Syntax and Semantics of Type Assignment Systems

Hirofumi Yokouchi
Department of Computer Science, Gunma University
1-5-1, Tenjin, Kiryu, Gunma, 376, Japan
yokouchi@cs.gunma-u.ac.jp

Abstract

This paper gives a survey on syntax and semantics for type assignment systems, with a special attention to semantic completeness of the systems. Starting with the most basic system with function types only, it introduces polymorphic types, intersection types, union types, and existential type quantifier in a step-by-step manner. It also provides several sequent-style formulations of type assignment systems. With the sequent calculi, it shows the properties of type assignment systems concerning the completeness and the conservativity of various systems.

1 Introduction

There are two ways to introduce types into lambda calculus. The one is due to Curry [10], and the system is called *type assignment system.* The original system was defined with combinatory logic instead of lambda calculus, but it was modified in a natural way to the lambda calculus [11, 12]. The other is due to Church [8], and the system is called *explicitly-typed lambda calculus.* In this survey, we focus on type assignment systems, and we present how naturally we can define semantics and syntax for them with a special attention to the semantic completeness. A criterion of the naturality is whether or not a syntactical system is complete for a semantics. There are a lot of variations and extensions for type assignment systems. In this survey, we mainly treat intersection and union types, and universal and existential quantifiers in addition to the basic systems with function types.

A type assignment system is designed on the operation of assigning a type to a type-free λ-term. In this survey, we write $M : \sigma$ for assigning a type σ to a type-free λ-term M. The expression $M : \sigma$ is called a statement, and we say that M has type σ or that σ is a type of M. The most basic type constructor is $\rightarrow$ for defining a function type, since the lambda calculus is an abstract theory of functions. In the type assignment system, we can deduce, for example,

$$\lambda x.x : \sigma \rightarrow \sigma$$

for any type σ.

The basic system with function types has been extended by introducing various type constructors. The first extension is polymorphic type introduced independently by Girard [19] and Reynolds [35]. The original system is based on the explicitly typed lambda calculus. The system due to Girard is called *system F*, and the one due to Reynolds is called *polymorphic typed lambda calculus* or *second-order typed lambda calculus*. As shown in the above example, the λ-term $\lambda x.x$ has type $\sigma \to \sigma$ for every type σ. To formalize this, we introduce type constructor $\forall$ and we write

$$\lambda x.x : \forall t.(t \to t).$$

A type with universal quantifier is called a polymorphic type. A function with a polymorphic type is called a polymorphic function. The concept of polymorphic functions is widely used in modern programming languages.

In type assignment systems, a type is interpreted as a set of values in a model of type-free lambda calculus. It is a natural attempt to extend types by introducing various set operations. One of such extensions is an intersection type introduced in [14]. We write $\sigma \wedge \tau$ for the type that denotes the intersection of the two sets corresponding to σ and τ. A theoretical motivation of intersection types is to extend the class of λ-terms that can have a type. For example, $\lambda x.xx$ cannot have any type in the system with function types only. With intersection type, it can have a type as:

$$\lambda x.xx : (\sigma \wedge (\sigma \to \tau)) \to \tau.$$

If variable x has type $\sigma \wedge (\sigma \to \tau)$, then it has both σ and $\sigma \to \tau$, and xx has type τ. Therefore, function $\lambda x.xx$ has type $(\sigma \wedge (\sigma \to \tau)) \to \tau$. More generally, it has been proved that every strongly normalizable λ-term has a type in the system with intersection types.

Intersection types can be used in designing a stronger type system in programming languages. A type denotes a set of values, so that a type can be regarded as a data-specification. It is natural to define various types that denote, for example, the set of all positive integer, the set of all even integers, and the set of all prime numbers. We may design a stronger type checking mechanism by using these types and the basic operations on types. For example, with types **Int** and **Boolean**, we can derive

$$\mathsf{iszero}(\mathsf{if}\ L\ \mathsf{then}\ 1\ \mathsf{else}\ -1) : \mathbf{Boolean}$$

if L is a term of type **Boolean**. This typing is carried out in any ordinary type system. Now we introduce type constants **NegInt**, **ZeroInt**, and **PosInt**, which denote the set of all negative integers, the singleton set $\{0\}$, and the set of all positive integers, respectively. Similarly we introduce **True** and **False** which denote the singleton sets of the truth value **true** and **false**, respectively. Then, we can assign a more refined type to the function iszero as:

$$\mathsf{iszero} : (\mathbf{NegInt} \to \mathbf{False}) \wedge (\mathbf{ZeroInt} \to \mathbf{True}) \wedge (\mathbf{PosInt} \to \mathbf{False}).$$

From this typing, we can derive

$$\mathsf{iszero}(\mathsf{if}\ L\ \mathsf{then}\ 1\ \mathsf{else}\ -1) : \mathbf{False}$$

for any λ-term L of type **Boolean**. This sample suggests the possibility that we may carry out a sort of program verification by means of type checking. More investigation on applications of intersection types to programming languages is presented in [34, 31, 32].

Existential quantifier and union are the companions to universal quantifier and intersection, respectively. Therefore it is natural to introduce existential quantifier types and union types. These types were discussed in [29, 24, 7]. It has been pointed that an existentially quantified type is related to the concept of abstract types [29]. A union type brings the flexibility of typing to programming languages. We write $\sigma \vee \tau$ for the type denoting the union of the two sets corresponding to σ and τ. Let us consider the following function $E(p)$ on the set $\mathbf{N}$ of natural numbers (including 0), which is taken from [30]:

$$E(p) = (\lambda f.\mathsf{succ}((ff)0))(\mathsf{if}\ \mathsf{iszero}(p)\ \mathsf{then}\ \lambda x.x\ \mathsf{else}\ \lambda yz.z).$$

Here succ is the successor function, which adds one, and iszero is the function checking whether the given value is 0 or not. If E is applied to value 0, then $\lambda x.x$ is passed to the function $\lambda f.\mathsf{succ}((ff)0)$ and the result of ff will be the identity function $\lambda x.x$. Therefore, the result of evaluating $E(0)$ will become 1. If E is applied to a positive integer, then $\lambda yz.z$ is passed to the function $\lambda f.\mathsf{succ}((ff)0)$ and the result will also be 1. There is no run-time error. However, the function E has no type in the type assignment systems with $\rightarrow$, $\forall$, and $\wedge$. The point is that (if iszero(p) then $\lambda x.x$ else $\lambda yz.z$) may yield two possible values $\lambda x.x$ and $\lambda yz.z$ according to the value of p, and these two λ-terms cannot have any common type. This is overcome by introducing union types. Define two types σ_1 and σ_2 as follows:

$$\begin{aligned} \sigma_1 &\equiv (\mathbf{N} \rightarrow \mathbf{N}) \wedge ((\mathbf{N} \rightarrow \mathbf{N}) \rightarrow (\mathbf{N} \rightarrow \mathbf{N})), \\ \sigma_2 &\equiv (\mathbf{N} \rightarrow \mathbf{N} \rightarrow \mathbf{N}) \wedge ((\mathbf{N} \rightarrow \mathbf{N} \rightarrow \mathbf{N}) \rightarrow (\mathbf{N} \rightarrow \mathbf{N})). \end{aligned}$$

Then, $\lambda x.x$ and $\lambda yz.z$ have the types σ_1 and σ_2, respectively, and therefore, (if iszero(p) then $\lambda x.x$ else $\lambda yz.z$) has the type $\sigma_1 \vee \sigma_2$. Moreover, if f has the type $\sigma_1 \vee \sigma_2$, then ff has type $\mathbf{N} \rightarrow \mathbf{N}$ and $(ff)0$ has type $\mathbf{N}$. Consequently, the function E can have type $\mathbf{N} \rightarrow \mathbf{N}$ if a union type is allowed.

Another aspect of type systems is the relationship with intuitionistic logic. It is known that there is a close correspondence between types and logical formulas in intuitionistic logic [11, 22, 13]. This correspondence is often called *Curry-Howard isomorphism* or *formulae-as-types isomorphism*. When we define various type systems, the known logical systems provide us with useful information. Indeed, most typ systems are designed on the analogy of natural deduction system for intuitionistic logic. However, it is remarkable that the exact Curry-Howard isomorphism does not hold for the systems with intersection or union types, although it holds for the systems with function types and universal quantifier types. Is is also possible to define type assignment systems based on Gentzen's sequent calculi. In this survey, after introducing type assignment systems in the ordinary style, we define them in the sequent-style and present analysis of type assignment systems by the sequent-style formulations.

In Section 2, we present the most basic system with function types only. In Section 3 through 5, we introduce polymorphic types, intersection types, union types and existential types, in a step-by-step manner. In Section 6, we introduce type assignment systems based on sequent calculi for logic, and in Section 7 we show several properties of type assignment with the sequent-style formulations.

The author would like to thank Masako Takahashi, Mariangiola Dezani-Ciancaglini, and anonymous referees.

2 Function Types

In this section we present the basic definitions and ideas of type assignment systems, introducing the systems which deal with only function types.

2.1 Preliminary Definitions

To begin with, we define preliminary notations on λ-terms and types. Let L, M, N, L_0, $L_1, \ldots$ stand for (type-free) λ-terms, and x, y, z, $x_0, x_1, \ldots$ stand for (term) variables. As a convention [5], two λ-terms are syntactically identified, whenever they are the same except for bound variables. When two λ-terms M and N are β-equivalent, we write $M \cong_\beta N$. When there is a β-reduction from M to N, we write $M \twoheadrightarrow_\beta N$. For a λ-term M, the set of all the free variables in M is denoted by $\mathrm{FV}(M)$. The term obtained from M by simultaneously substituting $N_1, \ldots, N_n$ for free variables $x_1, \ldots, x_n$ in M is denoted by $M[x_1, \ldots, x_n := N_1, \ldots, N_n]$. When a λ-term M has a normal form, the normal form of M is denoted by $\mathrm{norm}(M)$.

Given an infinite set of *type variables*, we define *types* by induction as follows:

- a type variable is a type,
- if σ and τ are types, then $(\sigma \to \tau)$ is a type.

Parentheses may be omitted when no confusion occurs. In particular, $\sigma_1 \to \cdots \to \sigma_n \to \tau$ stands for $(\sigma_1 \to (\cdots (\sigma_n \to \tau) \cdots))$. In this paper, σ, τ, ρ, $\sigma_0, \ldots$ stand for types, and s, t, u, $s_0, \ldots$ stand for type variables. For each pair of a λ-term M and a type σ, the expression $M : \sigma$ is called a (*typing*) *statement*. The λ-term M is called a *subject* of $M : \sigma$. A *basis* is a finite set of statements $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, where each subject x_i is a variable and no two statements have the same variable as subject. Let Γ, Δ, $\Gamma_0, \ldots$ stand for bases. A *sequent* is an expression of the form $\Gamma \rhd M : \sigma$.

2.2 Simple Semantics

The informal meaning of a type is a set of values, and a statement $M : \tau$ means that the value of M is contained in the set denoted by τ. In case M has free variables $x_1, \ldots, x_n$, the value of M depends on the value of these variables, and a sequent $x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \tau$ is intended to express that, if the value of each x_i is contained in the set denoted by σ_i, then the value of M is contained in the set

denoted by τ. To formalize this intuitive meaning of type assignment, we define the interpretations of λ-terms and types. For λ-terms, we use a λ-model, a model of type-free λ-calculus. The following is a standard definition of λ-model [5].

Definition (λ-models). A *λ-model* is a triple $\mathfrak{M} = (D, \cdot, [\![-]\!])$, where D is a nonempty set, $\cdot$ is a binary operation on D, and $[\![-]\!]$ is a mapping that assigns an element $[\![M]\!]_\xi \in D$ to each pair of a λ-term M and a function ξ, called a *term environment in* $\mathfrak{M}$, from the set of all variables to D. Moreover, a λ-model is required to satisfy the following conditions:

(1) $[\![x]\!]_\xi = \xi(x)$,

(2) $[\![MN]\!]_\xi = [\![M]\!]_\xi \cdot [\![N]\!]_\xi$,

(3) $[\![\lambda x.M]\!]_\xi \cdot a = [\![M]\!]_{\xi(x:=a)}$, where $\xi(x := a)$ is the term environment defined by $\xi(x := a)(x) = a$ and $\xi(x := a)(y) = \xi(y)$ if $x \not\equiv y$,

(4) if $\xi(x) = \xi'(x)$ for every $x \in \mathrm{FV}(M)$, then $[\![M]\!]_\xi = [\![N]\!]_{\xi'}$,

(5) if $[\![M]\!]_{\xi(x:=a)} = [\![N]\!]_{\xi(x:=a)}$ for every $a \in D$, then $[\![\lambda x.M]\!]_\xi = [\![\lambda x.N]\!]_\xi$.

Given a λ-model $\mathfrak{M} = (D, \cdot, [\![-]\!])$, a type is interpreted as a subset of D. It is reasonable to interpret type constructor $\to$ by

$$[A \to_S B] = \{\, d \in D \mid d \cdot a \in B \text{ for every } a \in A \,\}.$$

The semantics with this interpretation of function types is called *simple semantics*. The interpretation of types and the validity of sequents are defined as follows.

Definition (Interpretation of types in simple semantics). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be a λ-model. Let ν be a mapping, called a *type environment* over $\mathfrak{M}$, that assigns to each variable x, a subset $\nu(x) \subseteq D$. For each type σ, we define $[\![\sigma]\!]_\nu \subseteq D$ by induction as follows:

(1) $[\![t]\!]_\nu = \nu(t)$,

(2) $[\![\rho \to \tau]\!]_\nu = [\, [\![\rho]\!]_\nu \to_S [\![\tau]\!]_\nu \,]$,

Definition (Validity of sequents). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be an λ-model, ξ a term environment in $\mathfrak{M}$, and ν a type environment over $\mathfrak{M}$. A statement $M : \sigma$ is said to be *valid* in $(\mathfrak{M}, \xi, \nu)$ if and only if $[\![M]\!]_\xi \in [\![\sigma]\!]_\nu$. A sequent $\Gamma \rhd M : \sigma$ is said to be *valid* in $(\mathfrak{M}, \xi, \nu)$ if and only if either $\xi(x) \notin [\![\rho]\!]_\nu$ for some $x : \rho$ in Γ or $[\![M]\!]_\xi \in [\![\sigma]\!]_\nu$. Moreover, $\Gamma \rhd M : \sigma$ is said to be *valid* in $\mathfrak{M}$ if and only if it is valid in $(\mathfrak{M}, \xi, \nu)$ for every pair of term environment ξ and type environment ν.

2.3 Type Inference Rules

The inference system $\mathbf{T}_{\to}$ on sequents is defined by the following axioms and rules:

(Var) $\quad \Gamma \rhd x : \sigma \qquad (x : \sigma \in \Gamma)$

$(\to \text{I})$ $\quad \dfrac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x.M : \sigma \to \tau}$

$(\to \text{E})$ $\quad \dfrac{\Gamma \rhd M : \sigma \to \tau \qquad \Gamma \rhd N : \sigma}{\Gamma \rhd MN : \tau}$

In rule $(\to \text{I})$, $\Gamma, x : \sigma$ denotes the basis $\Gamma \cup \{x : \sigma\}$ provided no statements in Γ have the variable x as subject.

In this paper, we define a number of type assignment systems. In general, when a sequent $\Gamma \rhd M : \sigma$ can be derived in a system T, we say that $\Gamma \rhd M : \sigma$ is *derivable* in T, and write $T \vdash \Gamma \rhd M : \sigma$. Similarly, when a sequent $\Gamma \rhd M : \sigma$ cannot be derived in a system T, we say that $\Gamma \rhd M : \sigma$ is *underivable* in T, and write $T \nvdash \Gamma \rhd M : \sigma$. When no confusion occurs, we sometimes write $\vdash \Gamma \rhd M : \sigma$ and $\nvdash \Gamma \rhd M : \sigma$ by omitting the system name.

The following definitions are standard ones for inference systems.

Definition. Let T be a type assignment system, and (R) a rule of the form:

$$\frac{S_1 \quad \ldots \quad S_n}{S}$$

where $S_1, \ldots, S_n$, S are sequents.

(i) The rule (R) is said to be *admissible* in T when, if $T \vdash S_i$ for every i $(1 \leq i \leq n)$, then $T \vdash S$.

(ii) The rule (R) is said to be *derived* in T when there exists a derivation from S_i $(1 \leq i \leq n)$ to S with the axioms and rules in T.

(iii) The rule (R) is said to be *sound* in a λ-model $\mathfrak{M}$ when, if every S_i is valid in $\mathfrak{M}$, then S is valid in $\mathfrak{M}$.

For example, the rules $(\to \text{I})$ and $(\to \text{E})$ are sound in all λ-models. Moreover (Var) is valid in all λ-models, and therefore, we have the soundness theorem of $\mathbf{T}_{\to}$: if $\mathbf{T}_{\to} \vdash \Gamma \rhd M : \sigma$, then $\Gamma \rhd M : \sigma$ is valid in all λ-models. The following rule (Eq_β) is also sound in all λ-models:

(Eq_β) $\quad \dfrac{\Gamma \rhd M : \sigma}{\Gamma \rhd N : \sigma} \qquad (M \cong_\beta N)$

However, (Eq_β) is not admissible nor derivable in $\mathbf{T}_{\to}$. The following rules are admissible in $\mathbf{T}_{\to}$, but they are not derived in $\mathbf{T}_{\to}$:

(Weakening) $\quad \dfrac{\Gamma \rhd M : \tau}{\Gamma, x : \sigma \rhd M : \tau}$

$$\text{(Strengthening)}\quad \frac{\Gamma, x:\sigma \rhd M:\tau}{\Gamma \rhd M:\tau} \qquad (x \notin \mathrm{FV}(M))$$

$$\text{(Simple)}\quad \frac{\Gamma, x:\sigma \rhd Mx:\tau}{\Gamma \rhd M:\sigma\to\tau} \qquad (x \notin \mathrm{FV}(M))$$

These three rules are essentially used in the proof of the completeness theorem later.

We show two syntactical properties of type assignment systems. These two theorems are the most basic properties concerning the derivability, and they are satisfied by almost all type assignment systems in addition to $\mathbf{T}_{\to}$. For the proof, see, ex., [3].

Theorem 2.3.1 (Strong normalization). *If* $\mathbf{T}_{\to} \vdash \Gamma \rhd M:\sigma$, *then* M *is strongly normalizable. Namely, there is no infinite 1-step* β*-reduction sequence starting from* M.

Theorem 2.3.2 (Subject reduction). *If* $\mathbf{T}_{\to} \vdash \Gamma \rhd M:\sigma$ *and* $M \twoheadrightarrow_\beta N$, *then* $\mathbf{T} \vdash \Gamma \rhd N:\sigma$.

2.4 Completeness Theorem

The system $\mathbf{T}_{\to}$ is not semantically complete by a simple reason. For example, $\rhd (\lambda xy.y)((\lambda x.xx)(\lambda x.xx)) : \sigma\to\sigma$ is valid in all λ-models, but it is underivable in $\mathbf{T}_{\to}$. It is known that, if we add (Eq_β) to $\mathbf{T}_{\to}$, then the resulting system becomes complete for simple semantics.

Theorem 2.4.1 [20, 4] (Completeness for simple semantics). *A sequent is derivable in* $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$ *if and only if it is valid in all* λ*-models with respect to simple semantics.*

For the discussions later, we provide the proof here, following the one by Hindley [20]. Let Φ be an infinite set of statements whose subjects are pairwise distinct variables, such that Φ contains infinitely many statements $x_i:\sigma$ for every type σ. Let $\mathfrak{M}_0 = (D_0, \cdot, [\![-]\!])$ be the λ-model defined as follows:

(1) $D_0 = \{\,[M] \mid M \text{ is a } \lambda\text{-term}\,\}$,
where $[M]$ is the equivalence class containing M with respect to β-equality,

(2) $[M]\cdot[N] = [MN]$,

(3) $[\![M]\!]_\xi = [M[x_1,\ldots,x_n := N_1,\ldots,N_n]]$,
where $\mathrm{FV}(M) = \{x_1,\ldots,x_n\}$ and $\xi(x_i) = [N_i]$ for each i.

This λ-model is often called the *open term model.* Define term environment ξ_0 and type environment ν_0 by: $\xi_0(x) = [x]$, and

$$\nu_0(t) = \{\,[M] \mid \mathbf{T}_{\to} + (\mathrm{Eq}_\beta) \vdash \Phi \rhd M : t\,\},$$

where $\vdash \Phi \rhd M : t$ means that $\vdash \Delta \rhd M : t$ for some finite subset $\Delta \subseteq \Phi$. Then, by induction on σ, we show that

$$[\![M]\!]_{\xi_0} \in [\![\sigma]\!]_{\nu_0} \quad \text{if and only if} \quad \mathbf{T}_{\to} + (\mathrm{Eq}_\beta) \vdash \Phi \rhd M : \sigma.$$

When σ is a type variable, it is easily proved. Note that (Eq_β) is needed for the proof of the only-if part. Consider the case $\sigma \equiv \rho \to \tau$. Suppose $[\![M]\!]_{\xi_0} \in [\![\rho \to \tau]\!]_{\nu_0}$. By the construction of Φ, there is a variable $x : \rho \in \Phi$ with $x \notin \mathrm{FV}(M)$. By induction hypothesis, $[x] \in [\![\rho]\!]_{\nu_0}$, and therefore, by definition, $[Mx] \in [\![\tau]\!]_{\nu_0}$. By induction hypothesis, $\vdash \Phi \rhd Mx : \tau$. Let $\vdash \Delta \rhd Mx : \tau$ with $\Delta \subseteq \Phi$. It is easily proved that (Simple) and (Weakening) are admissible rules in $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$. If $x : \rho \in \Delta$, then $\vdash \Delta - \{x : \rho\} \rhd M : \rho \to \tau$ by (Simple). Otherwise, $\vdash \Delta \cup \{x : \rho\} \rhd Mx : \tau$ by (Weakening), and $\vdash \Delta \rhd M : \rho \to \tau$ by (Simple). Therefore, in both cases, we have $\vdash \Phi \rhd M : \rho \to \tau$. Conversely, suppose $\vdash \Phi \rhd M : \rho \to \tau$ and $[N] \in [\![\rho]\!]_{\nu_0}$. Then, by induction hypothesis, $\vdash \Phi \rhd N : \rho$, and therefore, $\vdash \Phi \rhd MN : \tau$. Using the induction hypothesis again, we have $[MN] \in [\![\tau]\!]_{\nu_0}$. Since N is arbitrary, we have $[M] \in [\![\rho \to \tau]\!]_{\nu_0}$.

Now we are ready to prove the completeness theorem. The if-part is straightforward by induction on the height of the derivation tree. For the only-if part, suppose $\Gamma \rhd M : \sigma$ is valid in all models. Take Φ such that $\Gamma \subseteq \Phi$. Then, $[\![M]\!]_{\xi_0} \in [\![\sigma]\!]_{\nu_0}$, and therefore, by the claim proved above, $\vdash \Phi \rhd M : \sigma$. Let $\vdash \Delta \rhd M : \sigma$ with $\Delta \subseteq \Phi$. Let Δ' be the basis obtained from Δ by removing all statements $x : \rho$ with $x \in \mathrm{FV}(M)$. It is easily proved that (Strengthening) is an admissible rules in $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$. Therefore, we have $\vdash \Delta' \rhd M : \sigma$ by (Strengthening), and $\vdash \Gamma \rhd M : \sigma$ by (Weakening). This completes the proof of the completeness theorem for $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$.

The proof suggests a stronger form of the completeness theorem.

Theorem 2.4.2 (Strong completeness). *There exist a λ-model $\mathfrak{M}$ and a type environment ν over $\mathfrak{M}$ such that, for every sequent $\Gamma \rhd M : \sigma$, the following two conditions are equivalent: (1) $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta) \vdash \Gamma \rhd M : \sigma$, and (2) $\Gamma \rhd M : \sigma$ is valid in $(\mathfrak{M}, \xi, \nu)$ for every term environment ξ in $\mathfrak{M}$.*

The strong completeness is proved by modifying the proof of Theorem 2.4.1. Let Φ, $\mathfrak{M}_0$, ξ_0, and ν_0 be the same as defined in the proof of Theorem 2.4.1. We show that Theorem 2.4.2 holds for $\mathfrak{M}_0$ and ν_0. The implication (1) $\Rightarrow$ (2) follows from the only-if part of Theorem 2.4.1. For (2) $\Rightarrow$ (1), suppose $\Gamma \rhd M : \sigma$ is valid in $(\mathfrak{M}_0, \xi, \nu_0)$ for every term environment ξ in $\mathfrak{M}_0$. For each $x : \sigma$ in Γ, we pick a variable $\hat{x}$ such that $\hat{x} \notin \mathrm{FTV}(M)$ and $\hat{x} : \sigma$ is contained in Φ, and we define $\hat{\Gamma}$ and $\hat{M}$ as the basis and the λ-term obtained from Γ and M, respectively, by replacing each x by $\hat{x}$. Then, $\hat{\Gamma} \rhd \hat{M} : \sigma$ is valid in $(\mathfrak{M}_0, \xi_0, \nu_0)$, and therefore, $[\![\hat{M}]\!]_{\xi_0} \in [\![\sigma]\!]_{\nu_0}$. By the claim proved in the proof of Theorem 2.4.1, $\vdash \Phi \rhd \hat{M} : \sigma$. In a similar way used in the proof of Theorem 2.4.1 we have $\vdash \hat{\Gamma} \rhd \hat{M} : \sigma$, and therefore, $\vdash \Gamma \rhd M : \sigma$. This completes the proof of Theorem 2.4.2.

2.5 F-Semantics

Another interpretation of function types is possible. A λ-model has the structure where a set of functions on the underlying set D is embedded into D. More precisely, given a λ-model $\mathfrak{M} = (D, \cdot, [\![-]\!])$, we define $[D \to D]$ of functions on D, and a pair of mappings $\phi : D \to [D \to D]$ and $\psi : [D \to D] \to D$ as follows:

$$[D \to D] = \{ f_a \mid a \in D \},$$

$$\phi(a) = f_a, \qquad \psi(f_a) = [\![\lambda x.zx]\!]_{\xi(z:=a)},$$

where f_a is the function on D defined by: $f_a(x) = a \cdot x$. Then, it is easily verified that $\phi \circ \psi = \mathrm{id}_{[D \to D]}$. In other words, $[D \to D]$ is embedded into D by ψ. Let F be the range of ψ, namely,

$$F = \{ [\![\lambda x.zx]\!]_{\xi(z:=a)} \mid a \in D \}.$$

We call F the *function kernel* of $\mathfrak{M}$.

The arguments above show that the "proper" representative of a function in $[D \to D]$ is an element of F. Therefore, it is reasonable to impose the restriction that a value in the interpretation of a function type should be in F. Taking this discussion into account, we can define another semantics of type assignment, called *F-semantics.*

Definition (Interpretation of types in F-semantics). The F-semantics is the same as semantics except that clause (2) for the interpretation of function types is replaced by:

$(2)_F$ $[\![\sigma \to \tau]\!]_\nu = [[\![\sigma]\!]_\nu \to_F [\![\tau]\!]\nu]$,

where $[A \to_F B] = \{d \in F \mid d \cdot a \in B \text{ for every } a \in A\}$.

It has been proved in [21] that $\mathbf{T}_\to + (\mathrm{Eq}_\beta)$ is also complete with respect to F-semantics. The proof is more difficult than the one for simple semantics. In case of F-semantics, we cannot use ξ_0 defined in the proof of Theorem 2.4.1. If $x : \sigma \to \tau$ is contained in Φ, then $\xi_0(x)$ must be contained in F. Therefore, we need careful treatment for defining ξ_0. For the detail, see [21]. When other type constructors are introduced, the model construction for the completeness becomes more complex.

2.6 Constants and Nonlogical Axioms

We can add constants and nonlogical axioms to $\mathbf{T}_\to$ as well as ordinary logical calculi. The definitions of term interpretation and type interpretation are modified in a trivial way when constants are added. Given $\mathfrak{M} = (D, \cdot, [\![-]\!])$, we introduce a valuation V that defines $V(c)$ for each term constant and $V(p)$ for each type

constant. The term interpretation $[\![-]\!]$ is extended to $[\![-]\!]^V$ for λ-terms with term constants by:

$$[\![M[x_1, \ldots, x_n := c_1, \ldots, c_n]]\!]^V_\xi = [\![M]\!]_{\xi(x_1,\ldots,x_n := V(c_1),\ldots,V(c_n))},$$

where M has no term constants. The type interpretation is also extended $[\![-]\!]^V$ for types with type constants by adding the clause: $[\![p]\!]^V_\nu = V(p)$ for type constant p. Let $\mathcal{A}$ be a set of statements whose subjects have no free variables and whose types contain no type variables. Then, the system $\mathbf{T}_\to$ is extended by adding following axioms:

$(\text{Axiom}_\mathcal{A})$ $\quad \Gamma \rhd M : \sigma \quad (M : \sigma \in \mathcal{A})$

The problem is the completeness of the resulting system. Namely, the following two are equivalent? (1) $\mathbf{T}_\to + (\text{Eq}_\beta) + (\text{Axiom}_\mathcal{A}) \vdash \Gamma \rhd M : \sigma$, and (2) $\Gamma \rhd M : \sigma$ is valid in all $(\mathfrak{M}, \xi, \nu)$ with constant valuation V such that every statement in $\mathcal{A}$ is valid in $\mathfrak{M}$ with V. The implication (1) $\Rightarrow$ (2) holds obviously. However, the converse does not generally hold. Recall the proof, described previously in Section 2.4, of the completeness theorem for the system without nonlogical axioms. In the proof, it is the key that the rule (Strengthening) is admissible in $\mathbf{T}_\to + (\text{Eq}_\beta)$. However, in case nonlogical axioms are added, this admissibility is no longer satisfied. For example, take $\mathcal{A} = \{\lambda x.c : p \to q\}$ with term constant c and type constants p and q. Then, we have $\mathbf{T}_\to + (\text{Axiom}_\mathcal{A}) + (\text{Eq}_\beta) \vdash x : p \rhd c : q$, while $\mathbf{T}_\to + (\text{Axiom}_\mathcal{A}) + (\text{Eq}_\beta) \nvdash \rhd c : q$.

A simple solution of this problem is to add (Strengthening) into the system. However, (Strengthening) is not a sound rule. For example, let $V(p) = \emptyset$ and $V(c) \notin V(q)$. Then, $\rhd \lambda x.c : p \to q$ and $x : p \rhd c : q$ are valid in the model with V, but $\rhd c : q$ is not valid. The point is that we allow a type to denote the empty set of values. If only nonempty types are considered, we have the completeness theorem.

Theorem 2.6.1 [27] (Strong completeness without empty types). *For every $\mathcal{A}$, there exist a λ-model $\mathfrak{M}$, valuation V, and type environment ν such that $V(p) \neq \emptyset$ for any type constant p, $\nu(t) \neq \emptyset$ for any type variable t, and the following two conditions are equivalent for every sequent $\Gamma \rhd M : \sigma$: (1) $\Gamma \rhd M : \sigma$ is derivable in $\mathbf{T}_\to + (\text{Axiom}_\mathcal{A}) + (\text{Strengthening}) + (\text{Simple}) + (\text{Eq}_\beta)$, and (2) for every term environment ξ, $\Gamma \rhd M : \sigma$ is valid in $(\mathfrak{M}, \xi, \nu)$ and V with respect to simple semantics.*

Corollary 2.6.2 (Completeness without empty types). *A sequent is derivable in $\mathbf{T}_\to + (\text{Axiom}_\mathcal{A}) + (\text{Strengthening}) + (\text{Simple}) + (\text{Eq}_\beta)$ if and only if it is valid in all $(\mathfrak{M}, \xi, \nu)$ and V with respect to simple semantics such that $V(p) \neq \emptyset$ for any type constant p and $\nu(t) \neq \emptyset$ for any type variable t.*

In the proof of Theorem 2.4.2, strong completeness of $\mathbf{T}_\to + (\text{Eq}_\beta)$, we used the fact that (Simple), (Strengthening), and (Weakening) are admissible. In Theorem 2.6.1, (Simple) and (Strengthening) are added to the system. The rule

(Weakening) is still admissible in $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Strengthening) + (Simple) + (Eq$_{\beta}$). Therefore, Theorem 2.6.1 follows immediately from the proof of Theorem 2.4.2.

We may expect that (Simple) can be replaced by the following simpler form:

$$(\eta) \qquad \frac{\Gamma \rhd \lambda x.Mx : \sigma}{\Gamma \rhd M : \sigma} \qquad (x \notin \mathrm{FV}(M))$$

Certainly (η) is admissible in $\mathbf{T}_{\to}$ + (Eq$_{\beta}$). However, this does not hold for the system with nonlogical axioms. For example, if we add nonlogical axiom $\lambda x.ax : p$ with term constant a and type constant p, then (η) is not admissible in the system $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Strengthening) + (Simple) + (Eq$_{\beta}$). Indeed, $\rhd \lambda x.ax : p$ is derivable, while $\rhd a : p$ is underivable.

Furthermore, (η) is not sound when constants are added. For example, let a be a term constant, and p a type constant. Given a λ-model $\mathfrak{M}$, we define a valuation V such that $V(p) = F$ and $V(a) \notin F$, where F be the function kernel of $\mathfrak{M}$. Then, $\rhd \lambda x.ax : p$ is valid in the model, since $[\![\lambda x.ax]\!]^V_\xi \in F$. On the other hand, $\rhd a : p$ is not valid, since $V(a) \notin F$. Therefore, (η) is not sound in the model $\mathfrak{M}$ with V.

2.7 Empty Types

We can extend $\mathbf{T}_{\to}$+(Axiom$_{\mathcal{A}}$) to handle empty types. We first present an example showing that $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Eq$_{\beta}$) is not complete. We take

$$\mathcal{A} = \{\lambda x.c : p_1 \to p_2,\ d : p_2 \to q,\ d : (p_1 \to p_3) \to q\}$$

with term constants c and d, and type constants p_1, p_2, p_3 and q. Then, $\rhd dc : q$ is valid in any models in which the statements in $\mathcal{A}$ are valid, while it is underivable in $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Eq$_{\beta}$). If $V(p_1) = \emptyset$, then $c : p_1 \to p_3$ is valid. Otherwise, $c : p_2$ is valid since $\lambda x.c : p_1 \to p_2$ is valid. Therefore, in both cases, $dc : q$ is valid. It is the key to this proof that we distinguish two cases whether $V(p_1) = \emptyset$. In the system $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Eq$_{\beta}$), however, we cannot express the hypothesis that a type is empty, and $dc : p \to q$ is underivable. A formal proof of the underivability will be presented in Section 7.3.

The discussion presented above suggests how the system should be extended. We introduce an expression of the form Empty(σ) into a basis. Intuitively, Empty(σ) means that σ is an empty type. The emptiness of types are handled by the following rules:

$$\text{(EmptyI)} \qquad \Gamma, x : \sigma, \mathrm{Empty}(\sigma) \rhd M : \tau$$

$$\text{(EmptyE)} \qquad \frac{\Gamma, x : \sigma \rhd M : \tau \qquad \Gamma, \mathrm{Empty}(\sigma) \rhd M : \tau}{\Gamma \rhd M : \tau}$$

Then, the resulting system satisfies the following completeness theorem.

Theorem 2.7.1 [27] (Completeness with empty types). *A sequent is derivable in* $\mathbf{T}_{\to}$ + (Axiom$_{\mathcal{A}}$) + (Simple) + (EmptyI) + (EmptyE) + (Eq$_{\beta}$) *if and only if it is valid in all* λ*-models* $\mathfrak{M}$ *and valuations* V *in which every statement in* $\mathcal{A}$ *is valid.*

Note that the system in Theorem 2.7.1 does not have (Strengthening), which is not generally sound in a model with empty types.

It is also remarkable that the strong completeness theorem like Theorem 2.6.1 does not hold for the system with (EmptyI) and (EmptyE). For example, let $\mathcal{A} = \{\lambda x.c : p \to q\}$ with term constant c and type constants p and q. Suppose that there exist a λ-model, valuation V, and type environment ν such that a sequent $\Gamma \rhd M : \sigma$ is derivable in $\mathbf{T}_{\to} + (\text{Axiom}_{\mathcal{A}}) + (\text{Simple}) + (\text{EmptyI}) + (\text{EmptyE}) + (\text{Eq}_{\beta})$ if and only if it is valid in $(\mathfrak{M}, \xi, \nu)$ and V for every ξ. Then, we have $V(p) = \emptyset$, since $\rhd \lambda x.c : p \to q$ is derivable and $\rhd c : q$ is not derivable. Therefore, $[\![p \to q]\!]^V_\xi$ contains all values, so that, by the assumption, every sequent of the form $\Gamma \rhd M : p \to q$ must be derivable. However, this is impossible. Consequently, the strong completeness theorem does not hold.

2.8 Curry-Howard Isomorphism

It is known that there is a close correspondence between types and logical formulas. If we erase any information on λ-terms in (var), ($\to$ I), and ($\to$ E) of $\mathbf{T}_{\to}$, then we get the following axioms and rules:

$$\Gamma \rhd \sigma \qquad (\sigma \in \Gamma)$$

$$\frac{\Gamma, \sigma \rhd \tau}{\Gamma \rhd \sigma \to \tau}$$

$$\frac{\Gamma \rhd \sigma \to \tau \qquad \Gamma \rhd \sigma}{\Gamma \rhd \tau}$$

If type constructor $\to$ is regarded as implication, then the resulting system is exactly a variant of natural deduction system for intuitionistic logic. In this setting, a type becomes a proposition, and a basis Γ becomes a finite set of propositions. The expression $\Gamma \rhd \sigma$ means that σ is deduced from the assumptions in Γ. We write $\mathbf{L}$ for this logical system. Let $\Gamma \rhd M : \sigma$ be derived in $\mathbf{T}_{\to}$. If we remove any information on λ-terms from the derivation tree for $\Gamma \rhd M : \sigma$ in $\mathbf{T}$, then the resulting tree becomes a derivation for $\Gamma^\circ \rhd \sigma$ in $\mathbf{L}$. Here Γ° is the set obtained from Γ by replacing each $x : \rho$ by ρ. Therefore, $\Gamma^\circ \rhd \sigma$ is provable in intuitionistic logic with implication only. Conversely, suppose $\Delta \rhd \tau$ is provable in intuitionistic logic. Then, there exist a λ-term M and a basis Γ such that $\Gamma^\circ = \Delta$ and $\Gamma \rhd M : \tau$ is derived in $\mathbf{T}_{\to}$. This is easily verified by induction on the derivation of $\Delta \rhd M : \tau$ in $\mathbf{L}$. A derivation tree for $\Gamma \rhd M : \sigma$ in $\mathbf{T}_{\to}$ is completely determined by the λ-term M. In fact, if M is, for instance, an application term, then the last applied rule should be ($\to$ E). Therefore, the λ-term M determines a proof of the proposition τ in $\mathbf{L}$. The correspondence between propositions (proofs) and types (λ-terms) is called Curry-Howard isomorphism. See [11, 22, 13]. It is summarized by the following theorem.

Theorem 2.8.1. (i) *If $\Gamma \rhd M : \sigma$ is derivable in $\mathbf{T}_{\to}$, then $\Gamma^\circ \rhd \sigma$ is derivable in $\mathbf{L}$.*

(ii) *If* $\Delta \rhd \tau$ *is derivable in* $\mathbf{L}$, *then there exists* $\Gamma \rhd M : \tau$ *derivable in* $\mathbf{T}_{\to}$ *such that* $\Gamma^{\circ} = \Delta$.

3 Polymorphic Types

In this section we introduce universal quantifier to the basic system, and we investigate how the definition of semantics is extended and how the completeness theorem is obtained.

3.1 Type Interpretation

We extend the definition of types by adding the following clause:

- if σ is a type and t is type variable, then $(\forall t\sigma)$ is a type.

We write $\forall t_1 \forall t_2 \ldots \forall t_n.\sigma$ for $(\forall t_1(\forall t_2(\ldots(\forall t_n(\sigma))\ldots)))$. Therefore, $\forall t.t \to t$ does not mean $((\forall t(t)) \to t)$ but $(\forall t(t \to t))$. This notation is used in Notation 4.1.6 of [3].

We prepair several notations for handling type variables in a polymorphic type. In a standard way, we define a free occurrence of type variables in a type σ and write $\mathrm{FTV}(\sigma)$ for the set of all the free type variables in σ. The type obtained from a type σ by simultaneously substituting types $\tau_1, \ldots, \tau_n$ for type variables $t_1, \ldots, t_n$ is denoted by $\sigma[t_1, \ldots, t_n := \tau_1, \ldots, \tau_n]$. For a basis Γ we define $\mathrm{FTV}(\Gamma)$ and $\Gamma[t_1, \ldots, t_n := \tau_1, \ldots, \tau_n]$ in a similar way.

Intuitively, the statement $M : \forall t.\sigma$ means that we have $M : \sigma$ for all possible values of the type variable t. For the interpretation of polymorphic types, it is needed to specify the range of possible values that type variables may have. Therefore, a model of the system with polymorphic types is defined as a pair of a λ-model $(D, \cdot, [\![-]\!])$ and a set $\mathcal{T}$ of subsets of D, where $\mathcal{T}$ is a range for type variables. We first define simple semantics.

Definition (Interpretation of types in simple semantics). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be a λ-model, $\mathcal{T}$ a set of subsets of D, ν a type environment that assigns to each variable, an element in $\mathcal{T}$. In simple semantics, for each type σ, we define the subset $[\![\sigma]\!]_{\nu} \subseteq D$ by induction as follows:

(1) $[\![t]\!]_{\nu} = \nu(t)$,

(2) $[\![\rho \to \tau]\!]_{\nu} = [[\![\rho]\!]_{\nu} \to_S [\![\tau]\!]_{\nu}]$,

(3) $[\![\forall t.\tau]\!]_{\nu} = \cap\{\, [\![\tau]\!]_{\nu(t:=P)} \mid P \in \mathcal{T} \,\}$.

Here $\nu(t := P)$ is the type environment defined by: $\nu(t := P)(t) = P$ and $\nu(t := P)(s) = \nu(s)$ for any type variable s other than t.

The pair $(\mathfrak{M}, \mathcal{T})$ is said to be a *model* for simple semantics if and only if $[\![\sigma]\!]_{\nu} \in \mathcal{T}$ for every pair of a type σ and a type environment ν in $\mathcal{T}$.

It is possible to define $\mathcal{T}$ as the set of all subsets of D. However, this causes a significant problem. In this interpretation, the type $\forall t.t$ is always interpreted as the empty set, and so any sequents of the form $\Gamma, x : \forall t.t \rhd M : \sigma$ are valid. It is reasonable for $\forall t.t$ to be empty in some models, but we also know that there exist models in which $\forall t.t$ is not empty. One of such models is found in [25, 24], in which a model is constructed with ideals on cpo's.

3.2 Type Inference Rules and Completeness

A polymorphic type is handled by the following pair of rules:

$$(\forall \text{ I}) \qquad \frac{\Gamma \rhd M : \sigma}{\Gamma \rhd M : \forall t.\sigma} \qquad (t \notin \mathrm{FTV}(\Gamma))$$

$$(\forall \text{ E}) \qquad \frac{\Gamma \rhd M : \forall t.\sigma}{\Gamma \rhd M : \sigma[t := \alpha]}$$

We define $\mathbf{T}_{\to\forall}$ as the system obtained from $\mathbf{T}_{\to}$ by adding ($\forall$ E) and ($\forall$ I) together with type constructor $\forall$.

We expect that $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$ is complete for simple semantics as well as $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$. However, this is not actually the case. For example,

$$x : s \to (\forall t.t) \rhd x : \forall t.s \to t$$

is valid in all models $(\mathfrak{M}, \mathcal{T})$ for simple semantics, while it is not derivable in $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$. It was essential to the proof of the completeness theorem for $\mathbf{T}_{\to}$ that (Simple) is admissible in $\mathbf{T}_{\to} + (\mathrm{Eq}_\beta)$. In $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$, however, (Simple) is not admissible. This suggests that we may obtain the completeness theorem if (Simple) is added. Indeed, we have the following completeness theorem.

Theorem 3.2.1 (Completeness of $\mathbf{T}_{\to\forall}$ for simple semantics). *A sequent is derivable in* $\mathbf{T}_{\to\forall} + (\mathrm{Simple}) + (\mathrm{Eq}_\beta)$ *if and only if it is valid in all models* $(\mathfrak{M}, \mathcal{T})$ *for simple semantics.*

Furthermore, $\mathbf{T}_{\to\forall}$ satisfies the strong completeness theorem as well as $\mathbf{T}_{\to}$. For the system with nonlogical axioms, we also have completeness results similar to Theorems 2.6.1, 2.6.2, and 2.7.1 for $\mathbf{T}_{\to}$.

3.3 F-Semantics

We define F-semantics by replacing the clause (2) in the definition of type interpretation by $(2)_F$ as defined in Section 2.5. However, the system $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$ is not complete for F-semantics. For example, $x : \forall t.t \rhd \lambda z.xz : \forall t.t$ is valid in all models for F-semantics, but it is underivable in $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$. The completeness problem for F-semantics is solved in [36] by introducing the following pair of rules:

$$(\mathrm{FI}) \qquad \frac{\Gamma \rhd M : \sigma \qquad \Gamma \rhd M : \rho_1 \to \rho_2}{\Gamma \rhd \lambda x.Mx : \sigma} \qquad (x \notin \mathrm{FV}(M))$$

$$\text{(FE)} \qquad \frac{\Gamma \rhd \lambda x.Mx : \sigma \qquad \Gamma \rhd M : \rho_1 \to \rho_2}{\Gamma \rhd M : \sigma} \qquad (x \notin \mathrm{FV}(M))$$

The definition of F-semantics requires that any λ-term M with function type should be interpreted as a value contained in the function kernel F of the given λ-model. The condition that $a \in F$ is expressed that $a = [\![\lambda x.zx]\!]_{\xi(z:=a)}$. Therefore, given M with function type, M and $\lambda x.Mx$ are interpreted as the same value in any models, so that M and $\lambda x.Mx$ have exactly the same types. This property of F-semantics is expressed by the rules (FI) and (FE). As the following completeness theorem shows, (FI) and (FE) are sufficient for characterizing F-semantics.

Theorem 3.2 [36] (Completeness of $\mathbf{T}_{\to\forall}$ for F-semantics). *A sequent is derivable in* $\mathbf{T}_{\to\forall}$ + (FI) + (FE) + (Eq_β) *if and only if it is valid in all models* $(\mathfrak{M}, \mathcal{T})$ *for F-semantics.*

3.4 Inference Semantics

As shown above, $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$ is not complete for simple semantics or F-semantics. The next problem is whether we can define another semantics that makes $\mathbf{T}_{\to\forall}$ + (Eq_β) complete. An answer is *inference semantics* proposed in [27].

Definition (Inference semantics). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be a λ-model and $[\![-]\!]$ a mapping that assigns $[\![\sigma]\!] \subseteq D$ to each type σ. The pair $(\mathfrak{M}, [\![-]\!])$ is said to be a *model for inference semantics* if and only if the following conditions are satisfied:

(1) $[[\![\rho]\!] \to_F [\![\tau]\!]] \subseteq [\![\rho \to \tau]\!] \subseteq [[\![\rho]\!] \to_S [\![\tau]\!]]$,

(2) $[\![\forall t.\tau]\!] = \cap\{\, [\![\tau[t := \alpha]]\!] \mid \alpha \text{ is a type} \,\}$.

Let ξ be a term environment. A sequent $\Gamma \rhd M : \sigma$ is said to be *valid* in $(\mathfrak{M}, [\![-]\!], \xi)$ if and only if either $[\![M]\!]_\xi \in [\![\sigma]\!]$ or $\xi(x) \notin [\![\rho]\!]$ for some $x : \rho$ in Γ. Furthermore, $\Gamma \rhd M : \sigma$ is said to be valid in $(\mathfrak{M}, [\![-]\!])$ if and only if it is valid in $(\mathfrak{M}, [\![-]\!], \xi)$ for all term environments ξ in $\mathfrak{M}$.

Note that (1) and (2) in inference semantics are not definitions but conditions. It is verified that $\mathbf{T}_{\to\forall}$ + (Eq_β) is complete for inference semantics. The proof is similar to the one for the completeness theorem with respect to simple semantics.

Theorem 3.4.1 (Completeness of $\mathbf{T}_{\to\forall}$ for inference semantics). *A sequent is derivable in* $\mathbf{T}_{\to\forall}$ + (Eq_β) *if and only if it is valid in all models for inference semantics.*

3.5 Coherent Semantics

It is remarkable that, in inference semantics, $[\![\sigma]\!] = [\![\sigma']\!]$ does not generally imply $[\![\tau[t := \sigma]]\!] = [\![\tau[t := \sigma']]\!]$. For example, in any model of inference semantics,

$[\![\forall t.t]\!] = [\![\forall s \forall t.t]\!]$, while $[\![(\forall t.t) \to \tau]\!] = [\![(\forall s \forall t.t) \to \tau]\!]$ does not generally hold. This is not natural because the value of a type is not determined the values of subexpressions of the type. To repair this unsatisfactory property, we introduce another semantics called *coherent semantics* [36].

Definition (Coherent semantics). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be a λ-model, $\mathcal{T}$ a set of subsets of D, $\to$ a binary operation on $\mathcal{T}$ that satisfies the condition that $[A \to_F B] \subseteq [A \to B] \subseteq [A \to_S B]$ for any $A, B \in \mathcal{T}$. For each pair of a type σ and a type environment ν in $\mathcal{T}$ we define $[\![\sigma]\!]_\nu \subseteq D$ by induction as follows:

(1) $[\![t]\!]_\nu = \nu(t)$,

(2) $[\![\rho \to \tau]\!]_\nu = [[\![\rho]\!]_\nu \to [\![\tau]\!]_\nu]$,

(3) $[\![\forall t.\tau]\!]_\nu = \cap\{\, [\![\tau]\!]_{\nu(t:=P)} \mid P \in \mathcal{T} \,\}$.

The triple $(\mathfrak{M}, \mathcal{T}, \to)$ is said to be a model for coherent semantics if and only if $[\![\sigma]\!]_\nu \in \mathcal{T}$ for every pair of type σ and type environment ν in $\mathcal{T}$.

It is remarkable that simple semantics and F-semantics can be regarded as special cases of coherent semantics. Indeed, let $(\mathfrak{M}, \mathcal{T})$ be a model for simple semantics. If we define binary operation $\to$ on $\mathcal{T}$ by $[A \to B] = [\![s \to t]\!]_{\nu(s:=A)(t:=B)}$, then $(\mathfrak{M}, \mathcal{T}, \to)$ becomes a model for coherent semantics.

The system $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$ is not complete for coherent semantics. For example, $x : (\forall t.t) \to \sigma \rhd x : (\forall s \forall t.t) \to \sigma$ is valid in all models for coherent semantics, while it is not derivable in $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$. The point is the fact that, in $\mathbf{T}_{\to\forall} + (\mathrm{Eq}_\beta)$, the types $\forall t.t$ and $\forall s \forall t.t$ are inhabited by exactly the same set of λ-terms, but the set of λ-terms with type $(\forall t.t) \to \sigma$ differs from that with type $(\forall s \forall t.t) \to \sigma$. We modify the system $\mathbf{T}_{\to\forall}$ so that the resulting system becomes complete for coherent semantics.

We define equivalence relation $\cong$ among types by the following axioms and rules:

(1) $\sigma \cong \sigma$,

(2) if $\sigma \cong \tau$, then $\tau \cong \sigma$,

(3) if $\rho \cong \sigma$ and $\sigma \cong \tau$, then $\rho \cong \tau$,

(4) if $\sigma \cong \sigma'$ and $\tau \cong \tau'$, then $\sigma \to \tau \cong \sigma' \to \tau'$,

(5) if $\sigma \cong \tau$, then $\forall t.\sigma \cong \forall t.\tau$,

(6) if $t \notin \mathrm{FTV}(\sigma)$, then $\forall t.\sigma \cong \sigma$,

(7) $\forall s \forall t.\sigma \cong \forall t \forall s.\sigma$.

With this equivalence relation we introduce the following rule:

$$\text{(EqType)} \qquad \frac{\Gamma \rhd M : \sigma}{\Gamma \rhd M : \tau} \qquad (\sigma \cong \tau)$$

This equivalence relation and rule are taken from [33]. If (EqType) is added, we have the completeness theorem for coherent semantics, which is proved in a similar way to the proof of the completeness for simple semantics.

Theorem 3.5.1 (Completeness of $\mathbf{T}_{\to\forall}$ for coherent semantics). *A sequent is derivable in* $\mathbf{T}_{\to\forall}$ + (EqType) + (Eq_β) *if and only if it is valid in all models for coherent semantics.*

3.6 Subtype Relation

Another topic on polymorphic types is the notion of subtype relation. We define preorder $\sqsubseteq$ among types by the following axioms and rules:

(1) $\sigma \sqsubseteq \sigma$,

(2) if $\rho \sqsubseteq \sigma$ and $\sigma \sqsubseteq \tau$, then $\rho \sqsubseteq \tau$,

(3) if $\sigma \sqsubseteq \sigma'$ and $\tau \sqsubseteq \tau'$, then $\sigma' \to \tau \sqsubseteq \sigma \to \tau'$.

(4) if $\sigma \sqsubseteq \tau$ and $t \notin \mathrm{FTV}(\sigma)$, then $\sigma \sqsubseteq \forall t.\tau$,

(5) $\forall t.\sigma \sqsubseteq \sigma[t := \alpha]$,

(6) if $t \notin \mathrm{FTV}(\sigma)$, then $\forall t.(\sigma \to \tau) \sqsubseteq \sigma \to (\forall t.\tau)$.

Intuitively, $\sigma \sqsubseteq \tau$ means that σ is a subset of τ. This subtype relation is introduced in [27].

We can show that the subtype relation $\sqsubseteq$ is characterized by $\mathbf{T}_{\to\forall}$ + (Simple). Namely, it is proved that $\mathbf{T}_{\to\forall}$ + (Simple) $\vdash x : \sigma \rhd x : \tau$ if and only if $\sigma \sqsubseteq \tau$. With the subtype relation we introduce the following rule:

$$\text{($\sqsubseteq$)} \qquad \frac{\Gamma \rhd M : \sigma}{\Gamma \rhd M : \tau} \qquad (\sigma \sqsubseteq \tau)$$

It is easily verified that $\mathbf{T}_{\to\forall}$ + (Simple) + (Eq_β) $\vdash \Gamma \rhd M : \sigma$ if and only if $\mathbf{T}_{\to\forall}$ + ($\sqsubseteq$) + (Eq_β) $\vdash \Gamma \rhd M : \sigma$. In the next section we define a subtype relation for the system with intersection types, and we demonstrate that it is used in a special model construction of type assignment.

4 Intersection Types

An intersection type is another extension to the type assignment systems. Roughly speaking, an intersection type is a finite fragment of universally-quantified type, and they have similar properties. The results on the completeness of the system with universal type quantifier are all extended to the system with intersection types.

4.1 Type Interpretation and Inference Rules

An intersection type is defined by adding the following clause to the definition of types:

- if σ and τ are types, then $(\sigma \wedge \tau)$ is a type.

We can naturally interpret intersection types in all styles of semantics introduced so far. In simple semantics, F-semantics, and coherent semantics, the type interpretation $[\![-]\!]$ is extended by adding the clause:

- $[\![\sigma \wedge \tau]\!]_\nu = [\![\sigma]\!]_\nu \cap [\![\tau]\!]_\nu$.

In inference semantics, we postulate the additional condition:

- $[\![\sigma \wedge \tau]\!] = [\![\sigma]\!] \cap [\![\tau]\!]$.

The intersection types are handled by the following rules:

$$(\wedge \text{ I}) \qquad \frac{\Gamma \rhd M : \sigma \qquad \Gamma \rhd M : \tau}{\Gamma \rhd M : \sigma \wedge \tau}$$

$$(\wedge \text{ E}) \qquad \frac{\Gamma \rhd M : \sigma \wedge \tau}{\Gamma \rhd M : \sigma} \qquad \frac{\Gamma \rhd M : \sigma \wedge \tau}{\Gamma \rhd M : \tau}$$

We define $\mathbf{T}_{\to\wedge}$ as the system obtained from $\mathbf{T}_{\to}$ by adding these rules together with type constructor $\wedge$. The resulting system satisfies the completeness theorems as well as $\mathbf{T}_{\to\forall}$.

Theorem 4.1.1 (Completeness of $\mathbf{T}_{\to\wedge}$ for simple semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge} + (\text{Simple}) + (\text{Eq}_\beta)$ *if and only if it is valid in all* λ*-models with respect to simple semantics.*

Theorem 4.1.2 (Completeness of $\mathbf{T}_{\to\wedge}$ for F-semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge} + (\text{FI}) + (\text{FE}) + (\text{Eq}_\beta)$ *if and only if it is valid in all* λ*-models with respect to F-semantics.*

Theorem 4.1.3 (Completeness of $\mathbf{T}_{\to\wedge}$ for inference semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge} + (\text{Eq}_\beta)$ *if and only if it is valid in all models for inference semantics.*

For coherent semantics, the equivalence relation $\cong$ among intersection types are defined by the following rules together with (1)–(4) presented in Section 3.5:

- if $\sigma \cong \sigma'$ and $\tau \cong \tau'$, then $\sigma \wedge \tau \cong \sigma' \wedge \tau'$,
- $\sigma \wedge \tau \cong \tau \wedge \sigma$,
- $\sigma \wedge \sigma \cong \sigma$.

Then, we have the completeness theorem for coherent semantics.

Theorem 4.1.4 (Completeness of $\mathbf{T}_{\to\wedge}$ for coherent semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge}$ + (EqType) + (Eq_β) *if and only if it is valid in all models for coherent semantics.*

These completeness theorems are easily proved by modifying the proofs of the same completeness theorem for $\mathbf{T}_{\to}$ or $\mathbf{T}_{\to\forall}$, and they also holds for the system $\mathbf{T}_{\to\wedge\forall}$ with both intersection types and polymorphic types.

We may hope that an intersection type corresponds to conjunction in Curry-Howard isomorphism. Indeed, if $\rhd M : \sigma$ is derivable in $\mathbf{T}_{\to\wedge}$, then σ is provable in intuitionistic logic. For example, $\rhd \lambda x.xx : \sigma \wedge (\sigma \to \tau) \to \tau$ is derivable in $\mathbf{T}_{\to\wedge}$, and $\sigma \wedge (\sigma \to \tau) \to \tau$ is provable in intuitionistic logic. However, the converse is not satisfied. For example, $\sigma \to \tau \to (\sigma \wedge \tau)$ is

provable in intuitionistic logic, but there is no λ-term with this type in $\mathbf{T}_{\to\wedge}$. The point is the form of rule ($\wedge$ I), in which the two upper sequents $\Gamma \rhd M : \sigma$ and $\Gamma \rhd M : \tau$ must have a common λ-term M. It is known that conjunction is corresponding to a product type instead of intersection type. The next problem is what kind of logic is corresponding to intersection type system. For this problem, a relationship with relevance logic is presented in [15].

4.2 The ω-type

The original system for intersection types has type constant ω. The ω-type is the universal type that includes all types. Precisely, the ω-type is interpreted in a λ-model $\mathfrak{M} = (D, \cdot, [\![-]\!])$ by:

- $[\![\omega]\!]_\nu = D$.

We define $\mathbf{T}_{\to\wedge\omega}$ as the system obtained from $\mathbf{T}_{\to\wedge}$ by adding the following rule together with the ω-type:

$$(\omega) \qquad \Gamma \rhd M : \omega$$

The system $\mathbf{T}_{\to\wedge\omega}$ has a nice property that (Eq_β) is an admissible rule. It is proved in [4] that (Eq_β) is an admissible rule in $\mathbf{T}_{\to\wedge\omega}$ and in $\mathbf{T}_{\to\wedge\omega}$ + (Simple). With this property, the completeness theorem of $\mathbf{T}_{\to\wedge\omega}$ becomes the following form.

Theorem 4.2.1 (Completeness of $\mathbf{T}_{\to\wedge\omega}$ for simple semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge\omega}$ + (Simple) *if and only if it is valid in all models with respect to simple semantics.*

Theorem 4.2.2 (Completeness of $\mathbf{T}_{\to\wedge\omega}$ for inference semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge\omega}$ *if and only if it is valid in all models with respect to inference semantics.*

For F-semantics, the pair of (FI) and (FE) is not strong enough to state the structure of F-semantics in case ω is added. For example,

$$z : t \wedge (\omega \to \omega \to \omega) \rhd \lambda xy.zxy : t$$

is valid in all models for F-semantics, but it is not derivable in $\mathbf{T}_{\to\wedge} + (\mathrm{FI}) + (\mathrm{FE}) + (\mathrm{Eq}_\beta)$. We need to strengthen (FI) and (FE) into the following forms:

(SFI) $$\frac{\Gamma \rhd \lambda x_1 \ldots x_n.M : \sigma \qquad \Gamma \rhd M : \rho \to \rho'}{\Gamma \rhd \lambda x_1 \ldots x_n x.Mx : \sigma}$$
($x \notin \mathrm{FV}(M)$ and none of $x_1, \ldots, x_n$ occur in Γ)

(SFE) $$\frac{\Gamma \rhd \lambda x_1 \ldots x_n x.Mx : \sigma \qquad \Gamma \rhd M : \rho \to \rho'}{\Gamma \rhd \lambda x_1 \ldots x_n.M : \sigma}$$
($x \notin \mathrm{FV}(M)$ and none of $x_1, \ldots, x_n$ occur in Γ)

It is proved in [36] that the resulting system satisfies the completeness theorem. See also [16] for the further discussion on F-semantics of the system with intersection types.

Theorem 4.2.3 [36] (Completeness of $\mathbf{T}_{\to\wedge\omega}$ for F-semantics). *A sequent is derivable in* $\mathbf{T}_{\to\wedge\omega}$ + (SFI) + (SFE) *if and only if it is valid in all models with respect to F-semantics.*

4.3 Filter Models

We define subtype relation $\sqsubseteq$ on the set of intersection types by the following axioms and rules:

(1) $\sigma \sqsubseteq \sigma$,

(2) if $\rho \sqsubseteq \sigma$ and $\sigma \sqsubseteq \tau$, then $\rho \sqsubseteq \tau$,

(3) if $\sigma \sqsubseteq \sigma'$ and $\tau \sqsubseteq \tau'$, then $\sigma' \to \tau \sqsubseteq \sigma \to \tau'$,

(4) if $\sigma \sqsubseteq \sigma'$ and $\tau \sqsubseteq \tau'$, then $\sigma \wedge \tau \sqsubseteq \sigma' \wedge \tau'$,

(5) $\sigma \wedge \tau \sqsubseteq \sigma$, $\sigma \wedge \tau \sqsubseteq \tau$,

(6) $(\rho \to \sigma) \wedge (\rho \to \tau) \sqsubseteq \rho \to (\sigma \wedge \tau)$,

(7) $\sigma \sqsubseteq \sigma \wedge \sigma$,

(8) $\sigma \sqsubseteq \omega$,

(9) $\omega \sqsubseteq \omega \to \omega$.

We find that this subtype relation is similar to the subtype relation on polymorphic types defined in Section 3.6. It is proved that $\sigma \sqsubseteq \tau$ if and only if $\mathbf{T}_{\to\wedge\omega}+(\text{Simple}) \vdash x : \sigma \rhd x : \tau$. Therefore, $\mathbf{T}_{\to\wedge\omega} + (\text{Simple})$ is equivalent to $\mathbf{T}_{\to\wedge\omega} + (\sqsubseteq)$, where $(\sqsubseteq)$ is the rule defined in Section 3.6 with the above subtype relation on intersection types.

Using the subtype relation we can provide another proof of the completeness theorems for simple semantics and inference semantics [4]. Here is an outline of the proof. We define a *filter* as a set F of types that satisfies the following conditions:

- $\omega \in F$,
- if $\sigma, \tau \in F$, then $\sigma \wedge \tau \in F$,
- if $\sigma \in F$ and $\sigma \sqsubseteq \tau$, then $\tau \in F$.

It is easily proved that the set D of all the filters becomes a λ-model, if we define $\cdot$ and $[\![-]\!]$ as follows:

$$F \cdot G = \{\, \tau \mid \sigma \to \tau \in F \text{ for some } \sigma \in G \,\},$$

$$[\![M]\!]_\xi = \{\, \tau \mid \mathbf{T}_{\to\wedge\omega} + (\text{Simple}) \vdash x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \tau \text{ for some } \sigma_i \in \xi(x_i)\ (1 \le i \le n) \,\},$$

where $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$. Let ν_0 be the type environment over the filter model define by

$$\nu_0(t) = \{\, F \mid F \text{ is a filter that contains } t \,\},$$

and ξ_Γ the term environment defined on a basis Γ by

$$\xi_\Gamma(x) = \{\, \sigma \mid \mathbf{T}_{\to\wedge\omega} \vdash \Gamma \rhd x : \sigma \,\}.$$

Then, it is easily verified that

$$\mathbf{T}_{\to\wedge\omega} + (\text{Simple}) \vdash \Gamma \rhd M : \sigma \text{ if and only if } [\![M]\!]_{\xi_\Gamma} \in [\![\sigma]\!]_{\nu_0}.$$

This fact implies the completeness theorem of $\mathbf{T}_{\to\wedge\omega} + (\text{Simple})$ for simple semantics. The point of this proof is the fact that the value of M is characterized by the set of the types inhabited by M.

We can also define another filter model that yields the completeness theorem of $\mathbf{T}_{\to\wedge\omega}$ for inference model. We first modify the subtype relation by removing (3), (6) and (9). The set of all the filters with respect to this modified subtype relation also becomes a λ-model. Define $[\![\sigma]\!] = \{F \mid F \text{ is a filter that contains } \sigma\}$. Then, the type interpretation $[\![-]\!]$ satisfies the conditions for inference semantics, and $\mathbf{T}_{\to\wedge\omega} \vdash \Gamma \rhd M : \sigma$ if and only if $[\![M]\!]_\xi \in [\![\sigma]\!]$ for every ξ such that $\xi(x) \in [\![\rho]\!]$ for any $x : \rho$ in Γ. This implies the completeness theorem of $\mathbf{T}_{\to\wedge\omega}$ for inference semantics.

4.4 Basic Syntactic Properties

We present two syntactic properties of the systems for intersection types. A theoretical motivation of intersection types is to characterize classes of λ-terms by types. In particular, strongly normalizable, normalizable, and solvable λ-terms are characterized by the shapes of their types, respectively. Here, a λ-term is said to be *normalizable* when there is a λ-term in normal form which is β-equivalent to the original λ-term. Similarly, a λ-term is said to be *solvable* when there is a λ-term of the form $\lambda x_1 \ldots, x_l.xN_1 \ldots N_m$ which is β-equivalent to the original λ-term. More rigorously, the original definition of solvable terms differs from ours, but it is proved that the original definition coincides with ours.

For each type σ, we define the notion of *positive* and *negative* occurrences of a type τ in σ, by simultaneous induction on the structure of σ:

- τ occurs positively in τ,
- τ occurs positively (negatively) in $\sigma_1 \to \sigma_2$ if either τ occurs negatively (positively) in σ_1 or τ occur positively (negatively) in σ_2,
- τ occurs positively (negatively) in $\sigma_1 \wedge \sigma_2$ if τ occurs positively (negatively) in σ_1 or σ_2.

Similarly we define the notion of *strongly positive* occurrence of τ in σ as follows:

- τ occurs strongly-positively in τ,
- τ occurs strongly-positively in $\sigma_1 \to \sigma_2$ if τ occur strongly-positively in σ_2,
- τ occurs strongly-positively in $\sigma_1 \wedge \sigma_2$ if τ occurs strongly-positively in σ_1 or σ_2.

For example, τ occurs positively in $(\tau \to \alpha) \to \beta$, negatively in $(\alpha \to \tau) \to \beta$, and strongly-positively in $\alpha \to \beta \to \tau$. Using the notations on occurrences in a type, we have a result on the characterization of three classes of λ-terms. The proof is presented in [23, 9]. See also [6].

Theorem 4.4.1. (i) *A λ-term M is strongly normalizable if and only if $\mathbf{T}_{\to\wedge} \vdash \Gamma \rhd M : \sigma$ for some Γ and σ.*

(ii) *A λ-term M is normalizable if and only if $\mathbf{T}_{\to\wedge\omega} \vdash \Gamma \rhd M : \sigma$ for some Γ and σ such that ω does not occur positively in σ and ω does not occur negatively in any types in Γ.*

(iii) *A λ-term M is solvable if and only if $\mathbf{T}_{\to\wedge\omega} \vdash \Gamma \rhd M : \sigma$ for some Γ and σ such that ω does not occur strongly-positively in σ.*

We next show the relationship between $\mathbf{T}_{\to\wedge}$ and $\mathbf{T}_{\to\forall}$. An intersection type is a finite approximation of a universally quantified type. Informally, a type $\forall t.\sigma$ is regraded as the type $\sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2] \wedge \ldots$ of infinite length, where the sequence $\alpha_1, \alpha_2, \ldots$ is an enumeration of all types. Therefore, for example,

$\sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2]$ is a finite approximation of $\forall t.\sigma$. Given an intersection type σ, we want to determine the universally quantified type of which σ is a finite approximation.

To do so we define subtype relation $\leq$ among the types of $\mathbf{T}_{\to\forall}$ as follows:

- $\sigma \leq \sigma$,
- if $\rho \leq \sigma$ and $\sigma \leq \tau$, then $\rho \leq \tau$,
- if $\sigma \leq \tau$ and $t \notin \mathrm{FTV}(\sigma)$, then $\sigma \leq \forall t.\tau$,
- $\forall t.\sigma \leq \sigma[t := \alpha]$.

The subtype relation $\leq$ is similar to $\sqsubseteq$ defined in Section 3.6 except that rules (3) and (6) for $\sqsubseteq$ are removed. The same relation $\leq$ can also be defined as follows:

$$\forall s_1 \ldots \forall s_l.\sigma \leq \forall t_1 \ldots \forall t_m.\sigma[s_1, \ldots, s_l := \tau_1, \ldots, \tau_l],$$

where $t_i \notin \mathrm{FTV}(\forall s_1 \ldots \forall s_l.\sigma)$ for every i $(1 \leq i \leq m)$. It is proved in [37] that every pair of types σ and τ in $\mathbf{T}_{\to\forall}$ has a greatest lower bound $\sigma \sqcap \tau$ with respect to $\leq$. For each type σ of $\mathbf{T}_{\to\wedge}$ we define the type $\mathrm{tr}(\sigma)$ of $\mathbf{T}_{\to\forall}$ as follows:

(1) $\mathrm{tr}(t) \equiv t$,

(2) $\mathrm{tr}(\sigma \to \tau) \equiv \mathrm{tr}(\sigma) \to \mathrm{tr}(\tau)$,

(3) $\mathrm{tr}(\sigma \wedge \tau) \equiv \mathrm{tr}(\sigma) \sqcap \mathrm{tr}(\tau)$.

Strictly speaking, this is not a correct definition, since a greatest lower bound of two types are not generally determined uniquely. Therefore, to define $\mathrm{tr}(\sigma \wedge \tau)$, we should choose a suitable representative of the equivalence class determined by the preorder $\leq$. For the exact definition, see [37]. We define a subsystem of $\mathbf{T}_{\to\wedge}$ such that $\mathbf{T}_{\to\forall}$ is essentially equivalent to the subsystem. Let $\mathbf{T}'_{\to\wedge}$ be the system obtained from $\mathbf{T}_{\to\wedge}$ by replacing ($\wedge$ I) by the following rule:

$$(\wedge\ \mathrm{I})' \quad \frac{\Gamma \rhd M : \rho \quad \Gamma \rhd M : \sigma \quad \Gamma \rhd M : \tau}{\Gamma \rhd M : \sigma \wedge \tau} \quad (\mathrm{tr}(\rho) \leq \mathrm{tr}(\sigma),\ \mathrm{tr}(\rho) \leq \mathrm{tr}(\tau))$$

In other words, the use of the rule ($\wedge$ I) is restricted in $\mathbf{T}'_{\to\wedge}$ by imposing the condition: $\Gamma \rhd M : \rho$ is derivable for some type ρ such that $\mathrm{tr}(\rho) \leq \mathrm{tr}(\sigma)$ and $\mathrm{tr}(\rho) \leq \mathrm{tr}(\tau)$.

We have the following theorem stating that $\mathbf{T}_{\to\forall}$ is embedded into $\mathbf{T}_{\to\wedge}$.

Theorem 4.4.2 [37] (Embedding $\mathbf{T}_{\to\forall}$ into $\mathbf{T}_{\to\wedge}$). (i) *If* $\mathbf{T}'_{\to\wedge} \vdash \Gamma \rhd M : \sigma$, *then* $\mathbf{T}_{\to\forall} \vdash \mathrm{tr}(\Gamma) \rhd M : \mathrm{tr}(\sigma)$.

(ii) *If* $\mathbf{T}_{\to\forall} \vdash \Delta \rhd M : \tau$, *then* $\mathbf{T}'_{\to\wedge} \vdash \Gamma \rhd M : \sigma$ *for some* Γ *and* σ *such that* $\Delta \cong \mathrm{tr}(\Gamma)$ *and* $\tau \cong \mathrm{tr}(\sigma)$, *where* $\cong$ *is the equivalence relation defined in Section 3.5.*

In the theorem, $\mathrm{tr}(\Gamma)$ stands for the basis obtained from Γ by replacing each $x : \rho$ by $x : \mathrm{tr}(\rho)$, and $\Delta \cong \mathrm{tr}(\Gamma)$ means that, if $x : \rho$ in Δ $(\mathrm{tr}(\Gamma))$, then $x : \rho' \in \mathrm{tr}(\Gamma)$ (Δ) for some ρ' such that $\rho \cong \rho'$.

5 Union and Existential Quantifier

We introduce union and existential quantifier, which are the dual constructors of intersection and universal quantifier, respectively. We show that the system with union types and/or existential types has very different characteristics from the system with intersection and/or universal quantifier.

5.1 Type Interpretation and Inference Rules

A union type and existential type are defined by the following clauses:

- if σ and τ are types, then $(\sigma \vee \tau)$ is a type,
- if t is a type variable and σ is a type, then $(\exists t\sigma)$ is a type.

For example, $(\forall s(\exists t(\sigma))$ is abbreviated to $\forall s\exists t.\sigma$. The type union and existential type quantifier are interpreted as union of two sets and union of infinitely many sets, respectively. Formally, the type interpretation in simple semantics, F-semantics, and coherent semantics is extended for union and existential quantifier by:

- $[\![\sigma \vee \tau]\!]_\nu = [\![\sigma]\!]_\nu \cup [\![\tau]\!]_\nu$,
- $[\![\exists t.\sigma]\!]_\nu = \cup\{\, [\![\sigma]\!]_{\nu(t:=P)} \mid P \in \mathcal{T} \,\}$.

For inference semantics, the following conditions are added:

- $[\![\sigma \vee \tau]\!] = [\![\sigma]\!] \cup [\![\tau]\!]$,
- $[\![\exists t.\sigma]\!] = \cup\{[\![\sigma[t := \alpha]]\!] \mid \alpha \text{ is a type}\}$.

The type union and existential type quantifier are handled by the following rules:

$$(\vee \text{ I}) \quad \frac{\Gamma \rhd M : \sigma}{\Gamma \rhd M : \sigma \vee \tau} \qquad \frac{\Gamma \rhd M : \tau}{\Gamma \rhd M : \sigma \vee \tau}$$

$$(\vee \text{ E}) \quad \frac{\Gamma \rhd N : \sigma \vee \tau \qquad \Gamma, x : \sigma \rhd M : \rho \qquad \Gamma, x : \tau \rhd M : \rho}{\Gamma \rhd M[x := N] : \tau}$$

$$(\exists \text{ I}) \quad \frac{\Gamma \rhd M : \sigma[t := \alpha]}{\Gamma \rhd M : \exists t.\sigma}$$

$$(\exists \text{ E}) \quad \frac{\Gamma \rhd N : \exists t.\sigma \qquad \Gamma, x : \sigma \rhd M : \rho}{\Gamma \rhd M[x := N] : \rho} \qquad (t \notin \text{FTV}(\Gamma) \cup \text{FTV}(\rho))$$

It is easily verified that the inference rules are all sound for simple semantics, F-semantics, and inference semantics. However, we meet the difficulty in trying to prove the completeness theorem. We know two proofs of the completeness theorem without union types or existential type quantifier. The first one was presented in Section 2.4 for proving Theorem 2.4.1, and the other was presented with filter models in Section 4.3. It is difficult to extend either proof into the completeness of the system with union types. The key to the first proof was the equivalence:

$$\mathbf{T}_{\to} + (\mathrm{Eq}_{\beta}) \vdash \Phi \rhd M : \sigma \text{ if and only if } [\![M]\!]_{\xi_0} \in [\![\sigma]\!]_{\nu_0}$$

In case a union type is added, this equivalence is no longer satisfied. Indeed, if this equivalence were satisfied, then we could have the equivalence:

$$\vdash \Phi \rhd M : \rho \vee \tau \text{ if and only if either } \vdash \Phi \rhd M : \rho \text{ or } \vdash \Phi \rhd M : \tau.$$

However, this equivalence does not generally hold. For example, let $x : \sigma \vee \tau \in \Phi$. Then, $\Phi \rhd x : \sigma \vee \tau$ is derivable, but neither $\Phi \rhd x : \sigma$ nor $\Phi \rhd x : \tau$ is derivable. Similarly, the key to the proof with filter model was the equivalence:

$$\mathbf{T}_{\to\wedge\omega} + (\mathrm{Simple}) \vdash \Gamma \rhd M : \sigma \text{ if and only if } [\![M]\!]_{\xi_\Gamma} \in [\![\sigma]\!]_{\nu_0}.$$

In case union and intersection types are introduced, this does not hold. If it were satisfied, then we could have the equivalence:

$$\vdash \Gamma \rhd M : \rho \vee \tau \text{ if and only if either } \vdash \Gamma \rhd M : \rho \text{ or } \vdash \Gamma \rhd M : \tau.$$

This does not generally hold.

Another difficulty is that the system with intersection and union types does not satisfy the subject reduction theorem like Theorem 2.3.2. The following example is taken from [2]. We can deduce the type

$$(\sigma \to \sigma \to \tau) \wedge (\rho \to \rho \to \tau) \to (\mu \to (\sigma \vee \rho)) \to \mu \to \tau$$

both for $\lambda xyz.x(yz)(yz)$ and $\lambda xyz.x(Iyz)(Iyz)$, but this type can be deduced neither for $\lambda xyz.x(Iyz)(yz)$ nor for $\lambda xyz.x(yz)(Iyz)$, where $I \equiv \lambda x.x$. In turn, the strong normalization theorem like Theorem 2.3.1 is satisfied fortunately. For the proof, see [38, 17, 23].

5.2 A Completeness Result in a Special Setting

It is shown in [2] that the system with intersection and union types becomes complete under a special setting. We introduce a special axiom and consider models that satisfying this axiom. In this section, we consider only the system $\mathbf{T}_{\to\wedge\vee}$ with intersection and union types only.

First we should note that the following sequent is always valid in all models but it is not derivable in $\mathbf{T}_{\to\wedge\vee}$.

(Dist $\vee$) $\qquad x : \rho \wedge (\sigma \vee \tau) \rhd x : (\rho \vee \sigma) \vee (\rho \wedge \tau)$

This sequent expresses the distributive law of $\wedge$ over $\vee$. We adopt this sequent as an axiom.

We introduce another axiom. We define the predicate $\mathbf{P}$ on types as follows:

- $\mathbf{P}(t)$ is true for any type variable t,
- $\mathbf{P}(\sigma \to \tau)$ is true if and only if $\mathbf{P}(\tau)$ is true,
- $\mathbf{P}(\sigma \wedge \tau)$ is true if and only if both $\mathbf{P}(\sigma)$ and $\mathbf{P}(\tau)$ are true,

- $\mathbf{P}(\sigma \vee \tau)$ is false.

We introduce the following axiom with the predicate $\mathbf{P}$:

(DisjProp) $x : \rho \to (\sigma \vee \tau) \rhd x : (\rho \to \sigma) \vee (\rho \to \tau)$, provided $\mathbf{P}(\rho)$ is true

The predicate $\mathbf{P}$ is also defined with the notation of strongly positive occurrences defined in Section 4.4. Namely, $\mathbf{P}(\sigma)$ is true if and only if no types of the form $\rho \vee \tau$ occur strongly-positively in σ. In predicate logic, a formula corresponding to such a type is said to be Harrop, and the rule (DisjProp) can be viewed as "Extended Disjunction Property" for a Harrop formula.

There is a natural model that satisfies (DisjProp). Let $\mathfrak{M} = (D, \cdot, [\![-]\!])$ be a λ-model constructed from a cpo D, and ν a type environment such that $\nu(t)$ has a least element. It is easily verified that, for every ρ, if $\mathbf{P}(\rho)$ is true, then $[\![\rho]\!]_\nu$ has a least element. Therefore, if $\mathbf{P}(\rho)$ is true and $a \in [\![\rho \to (\sigma \vee \tau)]\!]_\nu$, then a is contained in either $[\![\rho \to \sigma]\!]_\nu$ or $[\![\rho \to \tau]\!]_\nu$. Namely, all the instances of (DisjProp) are valid in $\mathfrak{M}$ and ν.

Let $\mathbf{T}_{\to\wedge\vee}$ be the system that consists of basic rules for $\to$, $\wedge$, and $\vee$. We define $\sigma \simeq \tau$ if and only if $x : \sigma \rhd x : \tau$ and $x : \tau \rhd x : \sigma$ are both derivable in $\mathbf{T}_{\to\wedge\vee}$ + (Dist $\vee$) + (DisjProp) + (Simple). Then, every type of $\mathbf{T}_{\to\wedge\vee}$ has a normal form with respect to $\simeq$ in the following sense that, for every type σ, there exists a type $\mathsf{m}(\sigma)$ of the form $\sigma_1 \vee \ldots \vee \sigma_n$ such that $\sigma \simeq \mathsf{m}(\sigma)$ and each σ_i is a type without union. Furthermore, the system with intersection and union types is characterized by the system with intersection types. Strictly, $\mathbf{T}_{\to\wedge\vee}$ + (Dist $\vee$) + (DisjProp) + (Simple) $\vdash M : \sigma$ if and only if $\mathbf{T}_{\to\wedge}$ + (Simple) $\vdash \rhd M : \sigma_i$ for some σ_i, where $\mathsf{m}(\sigma) \equiv \sigma_1 \vee \ldots \vee \sigma_n$. By this fact, we can obtain a completeness result for the system with (Dist $\vee$) and (DisjProp). Namely, a sequent is derivable in $\mathbf{T}_{\to\wedge\vee}$ + (Dist $\vee$) + (DisjProp) + (Simple) + (Eq_β) if and only if it is valid in all $\mathfrak{M}$ and ν with respect to simple semantics such that the following condition is satisfied: if $\mathbf{P}(\rho)$ is true, then $[\![\rho \to (\sigma \vee \rho)]\!]_\nu = [\![(\rho \to \sigma) \vee (\rho \to \tau)]\!]_\nu$. For the details, see [2].

6 Sequent-Style Formulations

In this section we introduce another formulation of type assignment, based on sequent calculi for predicate logic.

6.1 Sequent Calculus TLK

In the previous sections, a sequent is defined as an expression of the form:

$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \rhd M : \tau.$$

Rigorously speaking, the left-hand side of $\rhd$ should be a set instead of sequence. We extend sequents so that any λ-term is allowed as a subject in the left-hand side

of the sequents. Furthermore, we allow the right-hand side of each sequent to have more than one statements. Therefore, the extended sequents are of the form:

$$L_1 : \sigma_1, \ldots L_l : \sigma_l \rhd M_1 : \tau_1, \ldots, M_m : \tau_m.$$

In the rest of the paper, Γ, Δ, Π, Λ, $\Gamma_0, \ldots$ stand for finite sequences of statements, and a sequent is an expression of the form $\Gamma \rhd \Delta$. The sequences Γ and Δ of the sequent $\Gamma \rhd \Delta$ are called *antecedent* and *succedent*, respectively, of the sequent. The axioms and rules are shown in Figure 1. They are defined on the analogy of those for LK. We call the resulting system TLK because it corresponds to LK.

The axioms and rules of TLK resemble those of LK for classical logic. However, this does not imply that TLK corresponds to LK in Curry-Howard isomorphism. If only implication is considered, then TLK exactly corresponds to intuitionistic logic instead of classical logic in Curry-Howard isomorphism. The point is the form of $(\to \mathrm{R})$. In TLK, $(\to \mathrm{R})$ has the condition $x \notin \mathrm{FV}(\Gamma) \cup \mathrm{FV}(\Delta)$, while there is no condition for the corresponding rule in LK.

The validity of a sequent is naturally defined in a model for simple semantics and F-semantics, respectively. Let $(\mathfrak{M}, \mathcal{T})$ be a model for simple semantics or F-semantics, ξ a term environment in $\mathfrak{M}$, and ν a type environment in $\mathcal{T}$. A sequent $\Gamma \rhd \Delta$ is said to be *valid* in $(\mathfrak{M}, \mathcal{T}, \xi, \nu)$ if and only if either $[\![L]\!]_\xi \notin [\![\sigma]\!]_\nu$ for some $L : \sigma$ in Γ, or $[\![M]\!]_\xi \in [\![\tau]\!]_\nu$ for some $M : \tau$ in Δ. Moreover, a sequent is said to be *valid* in $(\mathfrak{M}, \mathcal{T})$ if and only if it is valid in $(\mathfrak{M}, \mathcal{T}, \xi, \nu)$ for all pairs of ξ and ν.

Similarly the validity of a sequent is defined in a model for inference semantics. Let $(\mathfrak{M}, \mathcal{T}, [\![-]\!])$ be a model in inference semantics, and ξ a term environment in $\mathfrak{M}$. A sequent $\Gamma \rhd \Delta$ is said to be *valid* in $(\mathfrak{M}, \mathcal{T}, [\![-]\!], \xi)$ if and only if either $[\![L]\!]_\xi \notin [\![\sigma]\!]$ for some $L : \sigma$ in Γ, or $[\![M]\!]_\xi \in [\![\tau]\!]$ for some $M : \tau$ in Δ. Moreover, a sequent is said to be *valid* in $(\mathfrak{M}, \mathcal{T}, [\![-]\!])$ if and only if it is valid in $(\mathfrak{M}, \mathcal{T}, [\![-]\!], \xi)$ for all term environments ξ.

6.2 Completeness of TLK

We show the completeness theorems of TLK. According to the modification of the definition of sequents, the rules (Eq_β) and (Simple) for TLK are defined as follows:

$$(\mathrm{Eq}_\beta) \qquad \frac{\Gamma \rhd \Delta, M : \tau}{\Gamma \rhd \Delta, N : \tau} \qquad (M \cong_\beta N)$$

$$(\text{Simple}) \qquad \frac{x : \sigma, \Gamma \rhd \Delta, Mx : \tau}{\Gamma \rhd \Delta, M : \sigma \to \tau} \qquad (x \notin \mathrm{FV}(\Gamma) \cup \mathrm{FV}(\Delta))$$

Theorem 6.2.1 (Completeness of TLK for simple semantics). *A sequent is derivable in* TLK + (Simple) + (Eq_β) *if and only if it is valid in all models for simple semantics.*

We present the outline of the proof in order to compare it with the proof of completeness described in Section 2.4. A complete proof is presented in [40].

(Initial) $M : \sigma \rhd M : \sigma$

(Weakening) $\dfrac{\Gamma \rhd \Delta}{M : \sigma,\ \Gamma \rhd \Delta}$ $\qquad$ $\dfrac{\Gamma \rhd \Delta}{\Gamma \rhd \Delta,\ M : \sigma}$

(Contraction) $\dfrac{M : \sigma,\ M : \sigma,\ \Gamma \rhd \Delta}{M : \sigma,\ \Gamma \rhd \Delta}$ $\qquad$ $\dfrac{\Gamma \rhd \Delta,\ M : \sigma,\ M : \sigma}{\Gamma \rhd \Delta,\ M : \sigma}$

(Exchange) $\dfrac{\Gamma_1,\ M : \sigma,\ N : \tau,\ \Gamma_2 \rhd \Delta}{\Gamma_1,\ N : \tau,\ M : \sigma,\ \Gamma_2 \rhd \Delta}$ $\qquad$ $\dfrac{\Gamma \rhd \Delta_1,\ M : \sigma,\ N : \tau,\ \Delta_2}{\Gamma \rhd \Delta_1,\ N : \tau,\ M : \sigma,\ \Delta_2}$

(Cut) $\dfrac{\Gamma_1 \rhd \Delta_1,\ M : \sigma \qquad M : \sigma,\ \Gamma_2 \rhd \Delta_2}{\Gamma_1,\ \Gamma_2 \rhd \Delta_1,\ \Delta_2}$

($\wedge$ L) $\dfrac{M : \sigma,\ \Gamma \rhd \Delta}{M : \sigma \wedge \tau,\ \Gamma \rhd \Delta}$ $\qquad$ $\dfrac{M : \tau,\ \Gamma \rhd \Delta}{M : \sigma \wedge \tau,\ \Gamma \rhd \Delta}$

($\wedge$ R) $\dfrac{\Gamma \rhd \Delta,\ M : \sigma \qquad \Gamma \rhd \Delta,\ M : \tau}{\Gamma \rhd \Delta,\ M : \sigma \wedge \tau}$

($\vee$ L) $\dfrac{M : \sigma,\ \Gamma \rhd \Delta \qquad M : \tau,\ \Gamma \rhd \Delta}{M : \sigma \vee \tau,\ \Gamma \rhd \Delta}$

($\vee$ R) $\dfrac{\Gamma \rhd \Delta,\ M : \sigma}{\Gamma \rhd \Delta,\ M : \sigma \vee \tau}$ $\qquad$ $\dfrac{\Gamma \rhd \Delta,\ M : \tau}{\Gamma \rhd \Delta,\ M : \sigma \vee \tau}$

($\rightarrow$ L) $\dfrac{\Gamma_1 \rhd \Delta_1,\ N : \sigma \qquad MN : \tau,\ \Gamma_2 \rhd \Delta_2}{M : \sigma \rightarrow \tau,\ \Gamma_1,\ \Gamma_2 \rhd \Delta_1,\ \Delta_2}$

($\rightarrow$ R) $\dfrac{x : \sigma,\ \Gamma \rhd \Delta,\ M : \tau}{\Gamma \rhd \Delta,\ \lambda x.M : \sigma \rightarrow \tau}$ $\quad (x \notin \mathrm{FV}(\Gamma) \cup \mathrm{FV}(\Delta))$

($\forall$ L) $\dfrac{M : \sigma[t := \alpha],\ \Gamma \rhd \Delta}{M : \forall t.\sigma,\ \Gamma \rhd \Delta}$

($\forall$ R) $\dfrac{\Gamma \rhd \Delta,\ M : \sigma}{\Gamma \rhd \Delta,\ M : \forall t.\sigma}$ $\quad (t \notin \mathrm{FTV}(\Gamma) \cup \mathrm{FTV}(\Delta))$

($\exists$ L) $\dfrac{M : \sigma,\ \Gamma \rhd \Delta}{M : \exists t.\sigma,\ \Gamma \rhd \Delta}$ $\quad (t \notin \mathrm{FTV}(\Gamma) \cup \mathrm{FTV}(\Delta))$

($\exists$ R) $\dfrac{\Gamma \rhd \Delta,\ M[t := \alpha]}{\Gamma \rhd \Delta,\ M : \exists t.\sigma}$

Figure 1: The axioms and rules of TLK

We use a technique similar to a standard proof of the completeness theorem for predicate logic. In case of logic, given a formula unprovable, we construct a maximal consistent set of formulas and define a model in which the formula is not valid. For type assignment, we also define a notion similar to a maximal consistent set of formulas. Let Φ and Ψ be (possibly infinite) sets of statements. The pair (Φ, Ψ) is said to be *maximal consistent* if and only if the following two conditions are satisfied:

(1) for any pair of finite sequences $\Gamma \subseteq \Phi$ and $\Delta \subseteq \Psi$, the sequent $\Gamma \rhd \Delta$ is underivable in TLK + (Simple) + (Eq_β),

(2) every statement is contained in either Φ or Ψ.

Suppose that $\Gamma_0 \rhd \Delta_0$ is underivable in TLK + (Simple) + (Eq_β). Then, we can construct a maximal consistent pair (Φ, Ψ) such that $\Gamma_0 \subseteq \Phi$ and $\Delta_0 \subseteq \Psi$. Moreover we can choose (Φ, Ψ) so that the following conditions are satisfied:

(a) if $M : \sigma \to \tau \in \Psi$, then $N : \sigma \in \Phi$ and $MN : \tau \in \Psi$ for some λ-term N,

(b) if $M : \forall t.\sigma \in \Psi$, then $M : \sigma[t := \alpha] \in \Psi$ for some type α,

(c) if $M : \exists t.\sigma \in \Phi$, then $M : \sigma[t := \alpha] \in \Phi$ for some type α.

With (Φ, Ψ) we define a model $(\mathfrak{M}_0, \mathcal{T}_0, \xi_0, \nu_0)$ as follows:

- $\mathfrak{M}_0$ is the open term model,
- $\|\sigma\| = \{\, [M] \mid M : \sigma \in \Phi \,\}$,
  where $[M]$ is the equivalence class of M in $\mathfrak{M}_0$,
- $\mathcal{T}_0 = \{\, \|\sigma\| \mid \sigma \text{ is a type} \,\}$,
- $\xi_0(x) = [x]$, $\nu_0(t) = \|t\|$.

It is easily verified that, in $(\mathfrak{M}_0, \mathcal{T}_0, \xi_0, \nu_0)$, all statements in Φ are valid and no statements in Ψ are valid. More precisely, we have the following equivalence:

$$M : \sigma \in \Phi \quad \text{if and only if} \quad [M] \in [\![\sigma]\!]_{\nu_0},$$
$$M : \sigma \in \Psi \quad \text{if and only if} \quad [M] \notin [\![\sigma]\!]_{\nu_0}$$

Form this it follows that $\Gamma_0 \rhd \Delta_0$ is not valid in $(\mathfrak{M}_0, \mathcal{T}_0, \xi_0, \nu_0)$.

The point of the above proof is the fact that we can express negation of a statement in TLK. The antecedent of each sequent in TLK may have a statement of any form, and therefore, the negation of statement $M : \sigma$ can be expressed by $M : \sigma \rhd$. The systems $\mathbf{T}_x$ such as $\mathbf{T}_\to$, on other hand, do not have this property, since statements in the antecedent of each sequent is restricted to the form $z : \sigma$ in $\mathbf{T}_x$.

For F-semantics, it is not clear how TLK becomes complete. For inference semantics, it can be proved that TLK + (Eq_β) is complete. Indeed, we define

$[\![\sigma]\!] = \|\sigma\|$ in the construction of $(\mathfrak{M}_0, \mathcal{T}_0, \xi_0, \nu_0)$ in the proof of the completeness theorem for simple semantics. Then, it is proved that $[\![-]\!]$ satisfies the conditions of inference semantics. By definition, $\Gamma_0 \rhd \Delta_0$ is not valid in the model $(\mathfrak{M}_0, [\![-]\!])$ for inference semantics.

The completeness theorem is extended into the case where nonlogical axioms are added. The system TLK satisfies the following property similar to the deduction theorem for LK: For every set $\mathcal{A}$ of statements without free term variable or free type variable, if TLK+(Simple)+(Eq_β)+$(\mathrm{Axiom}_\mathcal{A}) \vdash \Gamma \rhd \Delta$, then TLK+(Simple)+$(\mathrm{Eq}_\beta) \vdash \Pi, \Gamma \rhd \Delta$ for some finite sequence $\Pi \subseteq \mathcal{A}$. Therefore, with Theorem 6.2.1, we obtain the completeness theorem for TLK with nonlogical axioms.

Theorem 6.2.2 (Completeness of TLK with nonlogical axioms). *A sequent is derivable in* TLK + (Simple) + (Eq_β) + $(\mathrm{Axiom}_\mathcal{A})$ *if and only if it is valid in all models* $(\mathfrak{M}, \mathcal{T})$ *with valuations* V *for simple semantics in which every statement in* $\mathcal{A}$ *is valid.*

It is worth comparing Theorem 6.2.2 and Theorem 2.7.1. Both theorems state the completeness for models that may have empty types. In Theorem 2.7.1, we adopt the rules (EmptyI) and (EmptyE) with new kinds of statements $\mathrm{Empty}(\sigma)$. There rules make it possible to draw a derivation by distinguish cases whether a type is empty or not. On the other hand, TLK has no special statements like $\mathrm{Empty}(\sigma)$, but it allows the right-hand of a sequent to have more than one statements. By this extension, we can draw a derivation without explicitly distinguishing cases whether a type is empty or not. Consider the example which was taken in Subsection 2.7. Namely, let $\mathcal{A} = \{\lambda x.c : p_1 \to p_2,\ d : p_2 \to q,\ d : (p_1 \to p_3) \to q\}$. Then, we can derive $\rhd dc : q$ in $\mathbf{T}_\to$ + $(\mathrm{Axiom}_\mathcal{A})$ + (Simple) + (EmptyI) + (EmptyE). Similarly we can derive

$$\lambda x.c : p_1 \to p_2,\ d : p_2 \to q,\ d : (p_1 \to p_3) \to q \rhd dc : q$$

in TLK + (Simple) + (Eq_β) as shown in Figure 2. This derivation is a good example for understanding the rules of TLK.

6.3 Variations of Sequent Calculi

The definition of TLK suggests that we can define another calculus TLJ based on LJ for institutionistic logic. The system TLJ is obtained from TLK by imposing the restriction that the succedent of a sequent is a sequence that consists of exactly one statement. With this restriction, (Exchange), (Weakening), or (Contraction) for succedent is no longer needed, and ($\to$ L), for example, becomes the following form:

$$\frac{\Gamma_1 \rhd N : \sigma \qquad LN : \tau, \Gamma_2 \rhd M : \rho}{L : \sigma \to \tau, \Gamma_1, \Gamma_2 \rhd M : \rho}$$

The other rules are modified in a similar way.

Furthermore, for each of TLK and TLJ, we define two variants in which the antecedent of a sequent is restricted.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda x.c)x : p_2 \rhd (\lambda x.c)x : p_2}}{x : p_1 \rhd x : p_1 \qquad (\lambda x.c)x : p_2 \rhd c : p_2}}{\lambda x.c : p_1 \to p_2,\ x : p_1 \rhd c : p_2}}{x : p_1,\ \lambda x.c : p_1 \to p_2 \rhd c : p_2}}{x : p_1,\ \lambda x.c : p_1 \to p_2 \rhd c : p_2,\ cx : p_3}}{\lambda x.c : p_1 \to p_2 \rhd c : p_2,\ c : p_1 \to p_3 \qquad dc : q \rhd dc : q}}{d : (p_1 \to p_3) \to q,\ \lambda x.c : p_1 \to p_2 \rhd c : p_2,\ dc : q}}{d : (p_1 \to p_3) \to q,\ \lambda x.c : p_1 \to p_2 \rhd dc : q,\ c : p_2 \qquad dc : q \rhd dc : q}}{d : p_2 \to q,\ d : (p_1 \to p_3) \to q,\ \lambda x.c : p_1 \to p_2 \rhd dc : q,\ dc : q}}{\vdots}}{\lambda x.c : p_1 \to p_2,\ d : p_2 \to q,\ d : (p_1 \to p_3) \to q \rhd dc : q}$$

Figure 2: A Derivation in TLK with (Simple) and (Eq_β)

(1) We impose the restriction that the antecedent of a sequent is a sequence of statements whose subjects are variables only. With this restriction, ($\to$ L) and (Cut) are replaced by the following rules ($\to$ L)* and (Cut)*, respectively:

$$(\to \mathrm{L})^* \qquad \dfrac{\Gamma_1 \rhd \Delta_1, N : \sigma \qquad z : \tau, \Gamma_2 \rhd \Delta_2}{x : \sigma \to \tau, \Gamma_1, \Gamma_2 \rhd \Delta_1, \Delta_2[z := xN]} \qquad (z \notin \mathrm{FV}(\Gamma_2))$$

$$(\mathrm{Cut})^* \qquad \dfrac{\Gamma_1 \rhd \Delta_1, M : \sigma \qquad x : \sigma, \Gamma_2 \rhd \Delta_2}{\Gamma_1, \Gamma_2 \rhd \Delta_1, \Delta_2[x := M]} \qquad (x \notin \mathrm{FV}(\Gamma_2))$$

The other rules are the same as the original ones except that we impose the restriction on the antecedent of each sequent appearing in the rules. The resulting systems for TLK and TLJ are named TLK1 and TLJ1, respectively.

(2) We impose the restriction that the antecedent of a sequent is a sequence of statements whose subjects are pairwise distinct variables. With this restriction, (Cut) and ($\to$ L) are replaced by (Cut)* and ($\to$ L)*. Furthermore, (Contraction) for antecedent is replaced by the following rule:

$$(\mathrm{Contraction})^* \qquad \dfrac{x : \sigma, y : \sigma, \Gamma \rhd \Delta}{z : \sigma, \Gamma \rhd \Delta[x, y := z, z]}$$

The other rules are the same as the original ones except that we impose the restriction on the antecedent of each sequent appearing in the rules. The resulting systems for TLK and TLJ are named TLK2 and TLJ2, respectively.

After all, we have obtained six systems TLK, TLK1, TLK2, TLJ, TLJ1, and TLJ2. These systems are summarized in Figure 3 with the basic relationship among

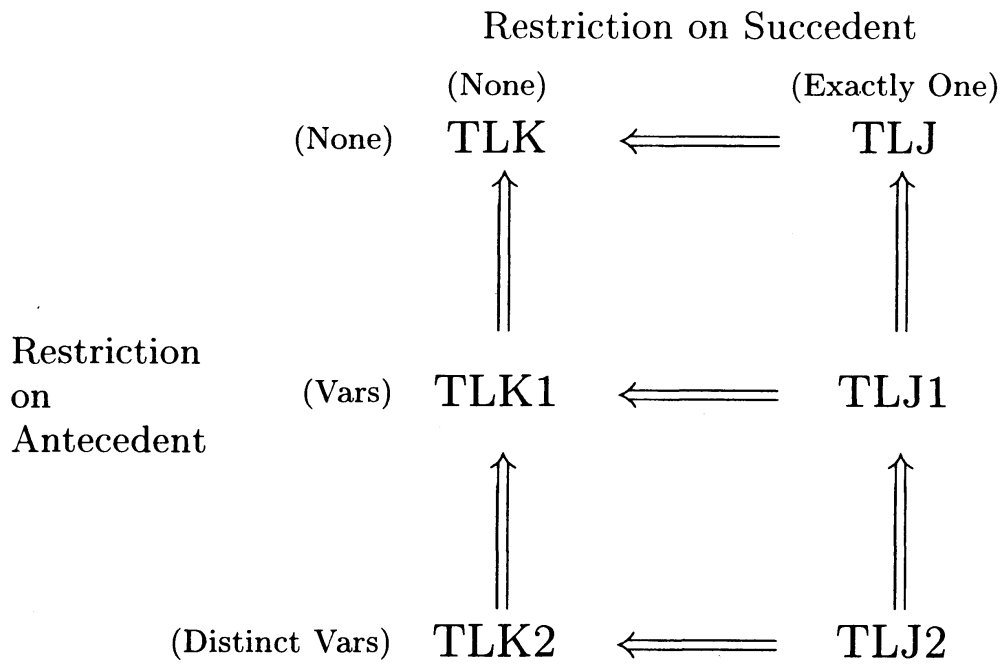

Figure 3: Variations of Sequent Calculi for Type Assignment

them. For instance, TLK $\Longleftarrow$ TLJ in Figure 3 shows that TLK is an extension of TLJ. Namely, every sequent in TLJ is also a sequent in TLK, and for every sequent S in TLJ, if S is derivable in TLJ, then so is in TLK. It follows immediately from definition that TLK is an extension of TLJ. The extensionalities for the other pairs are easily proved as well.

Another sequent-style formulation of type assignment is found in [1, 2, 7]. The system proposed in [1] is similar to TLJ except that subjects in the antecedent of a sequent are restricted to the form $xN_1 \ldots N_n$. The system proposed in [2, 7] is essentially equivalent to TLJ2, except that the rule (Contraction) is no explicitly treated in their system.

The systems defined in the previous sections are coincident with those defined in the formulations of TLJ2. In particular, let $\mathbf{T}_{\to\wedge\vee\forall\exists}$ be the system with all the type constructors $\to$, $\wedge$, $\vee$, $\forall$, and $\exists$ introduced in the previous sections. Then, it is easily proved that $\mathbf{T}_{\to\wedge\vee\forall\exists}$ is coincident with TLJ2. Note that the left-hand side of a sequent in $\mathbf{T}_{\to\wedge\vee\forall\exists}$ is a basis, a set of statements whose subjects are pairwise distinct variables. On the other hand, the antecedent of a sequent in TLJ2 is a sequence instead of a set. When a sequent in $\mathbf{T}_{\to\wedge\vee\forall\exists}$ is treated in TLJ2, we regard it as a sequent in TLJ2 by enumerating the elements in the basis of the sequent.

Theorem 6.3.1. *A sequent is derivable in* $\mathbf{T}_{\to\wedge\vee\forall\exists}$ *if and only if so is in* TLJ2. *This equivalence still holds even if* (Simple) *and/or* (Eq_β) *are added.*

By Theorem 6.2.1, TLK + (Simple) + (Eq_β) is complete for simple semantics. Therefore, if we clarify the relationship between TLK and TLJ2, then we may obtain the completeness theorem for $\mathbf{T}_{\to\wedge\vee\forall\exists}$. Actually, in Section 7.1, we show a completeness result by investigating the relationship among the six variants of TLK.

7 Analysis of Type Assignment by the Sequent-Style Formulations

In this section, using the sequent calculi introduced in Section 6, we show properties of type assignment systems including a completeness result.

7.1 Completeness of the System with Union and Existential Quantifier

We investigate the relationship among the six sequent calculi summarized in Figure 3, and we show a completeness result for the system with all type constructors including union and existential quantifier. In this section we treat only the systems with (Simple) and (Eq_β). So we write TLK* for the system obtained from TLK by adding (Simple) and (Eq_β). For the other calculi, we use similar notations. As shown in Section 6, TLK* is complete for models with respect to simple semantics, and TLJ2* is equivalent to $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$. Therefore, the completeness problem for $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$ is reduced to whether TLK* is equivalent to TLJ2*. This statement becomes more precise if we introduce a terminology on relationship between two systems.

Let T and T' be two type assignment systems. The system T' is said to be a conservative extension of T if and only if the following conditions are satisfied:

- every sequent in T is also a sequent in T',
- a sequent in T is derivable in T if and only if it is derivable in T'.

When T' is a conservative extension of T, we also say that T' is conservative over T. Using the terminology of conservative extension, our question is stated as follows: Whether TLK* is conservative over TLJ2*. However, the answer is no. For example, consider the following two sequents, which express the distributive laws of $\wedge$ over $\vee$ and over $\exists$, respectively:

(Dist $\vee$) $\quad x : (\sigma \vee \tau) \wedge \rho \rhd x : (\sigma \wedge \rho) \vee (\tau \wedge \rho)$

(Dist $\exists$) $\quad x : (\exists t.\sigma) \wedge \rho \rhd x : \exists t.\sigma \wedge \rho \qquad (t \notin \mathrm{FTV}(\rho))$

Of them, the former sequent has been introduced in Subsection 5.2. These two are derivable in TLJ1*, but neither is derivable in TLJ2*. A derivation for the former sequent in TLJ1* is shown in Figure 4. It is easily verified that, if we add these

$$\dfrac{\dfrac{\dfrac{\dfrac{x:\sigma \rhd x:\sigma}{x:\rho,\ x:\sigma \rhd x:\sigma} \quad \dfrac{x:\rho \rhd x:\rho}{x:\sigma,\ x:\rho \rhd x:\rho}}{x:\sigma,\ x:\rho \rhd x:\sigma \wedge \rho}}{x:\sigma,\ x:\rho \rhd x:(\sigma \wedge \rho) \vee (\tau \wedge \rho)} \quad \dfrac{\text{(similar)}}{x:\tau,\ x:\rho \rhd x:(\sigma \wedge \rho) \vee (\tau \wedge \rho)}}{x:\sigma \vee \tau,\ x:\rho \rhd x:(\sigma \wedge \rho) \vee (\tau \wedge \rho)}$$

$$\vdots$$

$$x:(\sigma \vee \tau) \wedge \rho \rhd x:(\sigma \wedge \rho) \vee (\tau \wedge \rho)$$

Figure 4: A derivation in TLJ1*

two sequents as axioms to TLJ2*, then TLJ1* is conservative over the resulting system.

The next problem is whether TLJ* is conservative over TLJ1*. The answer is still no. For example, consider

$$\begin{aligned} &x : \forall t.((\alpha \to \alpha \to \gamma) \wedge (\beta \to \beta \to \gamma)) \vee (\tau \to \sigma \to \gamma), \\ &y : \sigma \to (\exists t.\tau),\ y : \forall t.t \to t, \\ &z : \sigma,\ z : \forall t.\alpha \vee \beta \\ &\rhd x(yz)z : \gamma, \end{aligned}$$

where t occurs free in α, β, and τ, and it does not occur free in σ or γ. This sequent is derivable in TLJ*, but it is not generally derivable in TLJ2*. See Figure 5 for the derivation in TLJ*. Furthermore, TLK* is not conservative over TLJ*. For example,

$$x : \forall t.(t \to t) \vee s \rhd x : (\forall t.t \to t) \vee s$$

is derivable in TLK*, but it is underivable in TLJ*. See Figure 6 for the derivation in TLK*.

In these two counter examples against the conservativity, it is significant where type quantifiers occur in a type. In order to avoid these counter example, we define a class of sequents.

Definition (Stable sequents). Let $\mathbf{Ty}^+$ and $\mathbf{Ty}^-$ be the least sets of types such that the following conditions are satisfied:

(1) every type without type quantifier is contained in $\mathbf{Ty}^+$ and $\mathbf{Ty}^-$,

(2) if $\sigma,\ \tau \in \mathbf{Ty}^+$, then $\sigma \wedge \tau,\ \sigma \vee \tau \in \mathbf{Ty}^+$,

(3) if $\sigma,\ \tau \in \mathbf{Ty}^-$, then $\sigma \wedge \tau,\ \sigma \vee \tau \in \mathbf{Ty}^-$,

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \rho,\ yz : \tau,\ y : \forall t.t \to t,\ z : \sigma,\ z : \alpha \vee \beta \rhd x(yz)z : \gamma}\ D_1 \quad D_2}}{\vdots}}{yz : \tau,\ x : \forall t.\rho,\ y : \forall t.t \to t,\ z : \sigma,\ z : \forall t.\alpha \vee \beta \rhd x(yz)z : \gamma}}{z : \sigma \rhd z : \sigma \qquad yz : \exists t.\tau,\ x : \forall t.\rho,\ y : \forall t.t \to t,\ z : \sigma,\ z : \forall t.\alpha \vee \beta \rhd x(yz)z : \gamma}}{y : \sigma \to (\exists t.\tau),\ z : \sigma,\ x : \forall t.\rho,\ y : \forall t.t \to t,\ z : \sigma,\ z : \forall t.\alpha \vee \beta \rhd x(yz)z : \gamma}}{\vdots}}{x : \forall t.\rho,\ y : \sigma \to (\exists t.\tau),\ y : \forall t.t \to t,\ z : \sigma,\ z : \forall t.\alpha \vee \beta \rhd x(yz)z : \gamma}$$

where $t \notin \mathrm{FTV}(\sigma)$, $\rho \equiv ((\alpha \to \alpha \to \gamma) \wedge (\beta \to \beta \to \gamma)) \vee (\tau \to \sigma \to \gamma)$, and D_1 and D_2 are derivations for

$$x : (\alpha \to \alpha \to \gamma) \wedge (\beta \to \beta \to \gamma),\ yz : \tau,\ y : \forall t.t \to t,\ z : \sigma,\ z : \alpha \vee \beta \rhd x(yz)z : \gamma$$

and

$$x : \tau \to \sigma \to \gamma,\ yz : \tau,\ y : \forall t.t \to t,\ z : \sigma,\ z : \alpha \vee \beta \rhd x(yz)z : \gamma,$$

respectively.

Figure 5: A Derivation in TLJ*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : t \to t \rhd x : t \to t}}{x : t \to t \rhd x : t \to t,\ x : s}}{x : t \to t \rhd x : s,\ x : t \to t} \qquad \dfrac{}{x : s \rhd x : s}}{x : s \rhd x : s,\ x : t \to t}}{x : (t \to t) \vee s \rhd x : s,\ x : t \to t}}{x : \forall t.(t \to t) \vee s \rhd x : s,\ x : t \to t}}{x : \forall t.(t \to t) \vee s \rhd x : s,\ x : \forall t.t \to t}}{\vdots}}{x : \forall t.(t \to t) \vee s \rhd x : (\forall t.t \to t) \vee s}$$

Figure 6: A Derivation in TLK*

(4) if $\sigma \in \mathbf{Ty}^+$ and t is a type variable, then $\forall t.\sigma \in \mathbf{Ty}^+$,

(5) if $\sigma \in \mathbf{Ty}^-$ and t is a type variable, then $\exists t.\sigma \in \mathbf{Ty}^-$,

(6) if $\sigma \in \mathbf{Ty}^-$ and $\tau \in \mathbf{Ty}^+$, then $\sigma \to \tau \in \mathbf{Ty}^+$,

(7) if $\sigma \in \mathbf{Ty}^+$ and $\tau \in \mathbf{Ty}^-$, then $\sigma \to \tau \in \mathbf{Ty}^-$.

A statement $M : \sigma$ is said to be *stable* if and only if $\sigma \in \mathbf{Ty}^+$. A sequent $\Gamma \rhd \Delta$ is said to be *stable* if and only if $\sigma \in \mathbf{Ty}^-$ for every type σ of statements in Γ, and $\tau \in \mathbf{Ty}^+$ for every type τ of statements in Δ.

We can also redefine $\mathbf{Ty}^+$ and $\mathbf{Ty}^+$ with the notion of positive and negative occurrences defined in Section 4.4. Namely, $\mathbf{Ty}^+$ ($\mathbf{Ty}^-$) is the set of all types in which no type of the form $\forall t.\rho$ occurs negatively (positively) and no type of the form $\exists t.\rho$ occurs positively (negatively). Here are a few examples. If σ, τ_1, and τ_2 are types without type quantifier, then $L : \exists s.\sigma \rhd M : \forall t.((\exists u_1.\tau_1) \to (\forall u_2.\tau_2))$ is a stable sequent. On the other hand, neither $L : \forall s.\alpha \rhd M : \beta$ nor $L : \alpha \rhd M : \beta \to (\exists t.\gamma)$ is stable.

It can be proved that, if sequents are restricted to stable ones, then TLK* is conservative over TLJ* and TLJ* is conservative over TLJ1*. Furthermore we obtain the following theorem concerning the equivalence among the systems. For the proof, see [39, 38].

Theorem 7.1.1 (Equivalence among systems). *Let S be a stable sequent allowed in* TLJ2*. *Namely, the antecedent of S consists of statements whose subjects are distinct variables. Then, the derivabilities of S in the following six systems are all equivalent to each another:* TLK*, TLK1*, TLK2* + (Dist $\exists$), TLJ*, TLJ1*, TLK2* + (Dist $\vee$) + (Dist $\exists$), *and* $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$ + (Dist $\vee$) + (Dist $\exists$).

Form this theorem and the completeness of TLK*, we have the completeness of $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$ for stable sequents.

Theorem 7.1.2 (Completeness of $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$). *A stable sequent is derivable in* $\mathbf{T}^*_{\to\wedge\vee\forall\exists}$ + (Dist $\vee$) + (Dist $\exists$) *if and only if it is valid in all models with respect to simple semantics.*

7.2 Cut-Elimination

In the logical calculi LK and LJ, the cut-elimination theorem, due to Gentzen [18] plays an essential role for showing various properties of the calculi. Of the family of sequent calculi for type assignment, the systems TLK*, TLJ*, and TLJ2* enjoy the cut-elimination. Namely, if a sequent is derivable, then it can be derived without using (Cut) or (Cut)*. The proof is not very easy. It is difficult to use the original proof technique by Gentzen. This difficulty comes from the impredicativity of type quantifiers. For the detail, see [40, 38].

In this section we show two applications of the cut-elimination theorem. The first application is to prove the conservativity of the systems. Let T be either TLK*, TLJ*, or TLJ2*. For each nonempty set C of type constructors $\to$, $\wedge$, $\vee$, $\forall$, and $\exists$, we define T_C as the system obtained from T by imposing the restriction that types are constructed from type variables by applying only the type constructor in C. Then we have the following theorem concerning the conservativity.

Theorem 7.2.1. *Let T be either* TLK*, TLJ*, *or* TLJ2*. *Let A and B be nonempty sets of type constructors $\to$, $\wedge$, $\vee$, $\forall$, and $\exists$ such that $A \subseteq B$. If A does not contain $\forall$ or $\exists$, then T_B is a conservative extension of T_A.*

This theorem is an immediate conclusion from the cut-elimination theorem. Indeed, suppose that a sequent S of T_A is derivable in T_B. Then, there exists a cut-free derivation tree for S in T_B. All types occurring in the cut-free derivation tree are subexpressions of types in S. (In LK or LJ, the corresponding property is called subformula property.) Therefore, the derivation tree is allowed in T_A, so that S is derivable in T_A.

It is remarkable that Theorem 7.2.1 does not hold if we remove the restriction on A. Indeed, if A contains $\forall$ or $\exists$, then a cut-free derivation tree in T_A may have inferences by ($\forall$ L) or ($\exists$ R), and so the subformula property does not hold. However, if we restrict statements to stable ones defined in the theorem holds without the restriction on A. This follows from the fact that any cut-free derivation tree for a stable sequent has no inference of ($\forall$ L) or ($\exists$ R).

Theorem 7.2.2. *Let T be either* TLK*, TLJ*, *or* TLJ2*. *Let A and B be nonempty sets of type constructors $\to$, $\wedge$, $\vee$, $\forall$, and $\exists$ such that $A \subseteq B$. For every stable sequent S of T_A, the sequent S is derivable in T_A if and only if so is in T_B.*

The conservativity on TLJ2* is translated in $\mathbf{T}_{\to\wedge\vee\forall\exists}$. As shown in Theorem 6.3.1, TLJ2 and $\mathbf{T}_{\to\wedge\vee\forall\exists}$ are equivalent, and it can be extended for the systems with (Simple) and (Eq_β). The equivalence also holds for the subsystems of them. Therefore, the conservativity theorems presented above hold for the subsystems of $\mathbf{T}_{\to\wedge\vee\forall\exists}$ + (Simple) + (Eq_β).

Another application of the cut-elimination is to prove the underivability. In Section 7.1, we mention that the axioms (Dist $\vee$) or (Dist $\exists$) are not generally derivable in TLJ2*. By the cut-elimination of TLJ2*, we can prove this fact. Suppose that

$$x : (s \vee t) \wedge u \rhd x : (s \wedge u) \vee (s \wedge u)$$

is derivable. Then, it can be derived without (Cut)*. It is easily proved that there exists a cut-free derivation tree for that sequent in TLJ2. Let $\mathcal{P}$ be the shortest one of such cut-free derivation trees. Then, the rule applied at the last step of the derivation must be either ($\wedge$ L) or ($\vee$ R). Therefore, the upper sequent of the

inference at the last step is one of the following four sequents:

$$\begin{aligned} x : s \vee t &\rhd x : (s \wedge u) \vee (t \wedge u), \\ x : u &\rhd x : (s \wedge u) \vee (t \wedge u), \\ x : (s \vee t) \wedge u &\rhd x : s \wedge u, \\ x : (s \vee t) \wedge u &\rhd x : t \wedge u. \end{aligned}$$

However, none of these sequents is derivable in TLJ2. For instance, suppose the first sequent is derivable in TLJ2. There is a model in which S is not valid. In fact, we define ξ_0 and ν_0 so that $\xi_0(x) \in \nu_0(s)$, $\xi_0(x) \notin \nu_0(t)$, and $\xi_0(x) \notin \nu_0(u)$. Then, $x : s \vee t \rhd x : (s \wedge u) \vee (t \wedge u)$ is not valid in any model with ξ and ν. This contradicts the fact that all rules of TLJ2 are sound in all models.

7.3 Kripke-Semantics

The system TLJ is defined on the analogy with intuitionistic logic LJ. This suggests that we can define a Kripke model for type assignment. A typing statement is expressed by a logical formula with membership predicates for ground types. For example, the statement $M : u \to (s \vee t)$ is expressed by:

$$\forall x.((x \in u) \supset ((Mx \in s) \vee (Mx \in t))),$$

where $\in$, say $(-) \in s$, is the predicate for checking the membership for ground type s. If such a logical formula is interpreted in a model of classical logic, then we obtain the standard interpretation of types for simple semantics defined in Section 2. We can also interpret the above logical formula in a Kripke model for intuitionistic logic. If we define the interpretation of types directly, then we have a definition of Kripke models for type assignment. For the explicitly typed λ-calculus, Mitchell and Moggi [28] proposed a Kripke model, which follows from the investigation of empty types [26]. Our definition of Kripke-models is similar to their definition except that we treat type assignment version with type quantifiers.

In this section, we define Kripke models for simple semantics, and we show that TLJ satisfies the completeness theorem.

Definition (Kripke premodels). A *Kripke premodel* (for type assignment) is a triple $\mathcal{K} = (\mathcal{W}, \mathfrak{M}, \mathcal{T})$ such that:

(1) $\mathcal{W}$ is a nonempty set with preorder $\sqsubseteq$, whose elements are called *possible worlds*.

(2) $\mathfrak{M}$ is a family of λ-models $\mathfrak{M}_w = (D_w, \cdot_w, [\![-]\!]^w)$ indexed by possible worlds w such that, if $w \sqsubseteq w'$, then $\mathfrak{M}_w$ is a submodel of $\mathfrak{M}_{w'}$.

(3) $\mathcal{T}$ is a set of partial mappings, called *partial domains*, such that:

(a) the domain of P is a subset $\mathrm{Dom}(P) \subseteq \mathcal{W}$,

(b) P assigns to each $w \in \mathrm{Dom}(P)$, a subset $P[w] \subseteq D_w$,

(c) if $w \in \mathrm{Dom}(P)$ and $w \sqsubseteq w'$, then $w' \in \mathrm{Dom}(P)$ and $P[w] \subseteq P[w']$.

Definition (Type interpretation in Kripke premodels). Let $\mathcal{K} = (\mathcal{W}, \mathfrak{M}, \mathcal{T})$ be a Kripke premodel. A *type environment in* $\mathcal{T}$ is a mapping ν that assigns to each type variable t, a partial domain $\nu(t) \in \mathcal{T}$. For each pair of type σ and type environment ν we define the partial domain $[\![\sigma]\!]_\nu$ as follows:

- $\mathrm{Dom}([\![\sigma]\!]_\nu) = \bigcap\{\, \mathrm{Dom}(\nu(t)) \mid t \in \mathrm{FTV}(\sigma) \,\}$,
- for each $w \in \mathrm{Dom}([\![\sigma]\!]_\nu)$, the subset $[\![\sigma]\!]_\nu[w] \subseteq D_w$ is defined as follows:

(1) $[\![t]\!]_\nu[w] = \nu(t)[w]$,

(2) $[\![\rho \wedge \tau]\!]_\nu[w] = ([\![\rho]\!]_\nu[w]) \cap ([\![\tau]\!]_\nu[w])$,

(3) $[\![\rho \vee \tau]\!]_\nu[w] = ([\![\rho]\!]_\nu[w]) \cup ([\![\tau]\!]_\nu[w])$,

(4) $[\![\rho \to \tau]\!]_\nu[w]$
$= \{\, c \in D_w \mid \forall w' \sqsupseteq w \forall a \in [\![\rho]\!]_\nu[w'].(c \cdot_{w'} a \in [\![\tau]\!]_\nu[w']) \,\}$,

(5) $[\![\forall t.\tau]\!]_\nu[w]$
$= \{\, a \in D_w \mid \forall w' \sqsupseteq w \forall P \in \mathcal{T}(w' \in \mathrm{Dom}(P)).(a \in [\![\tau]\!]_{\nu(t:=P)}[w']) \,\}$,

(6) $[\![\exists t.\tau]\!]_\nu[w]$
$= \{\, a \in D_w \mid \exists P \in \mathcal{T}(w \in \mathrm{Dom}(P)).(a \in [\![\tau]\!]_{\nu(t:=P)}[w]) \,\}$.

Note that $[\![\sigma]\!]_\nu$ satisfies the condition of partial domains: if $w \in \mathrm{Dom}([\![\sigma]\!]_\nu)$ and $w \sqsubseteq w'$, then $w' \in \mathrm{Dom}([\![\sigma]\!]_\nu)$ and $[\![\sigma]\!]_\nu[w] \subseteq [\![\sigma]\!]_\nu[w']$.

Definition (Kripke models). A Kripke premodel $\mathcal{K} = (\mathcal{W}, \mathfrak{M}, \mathcal{T})$ is called a *Kripke model* if and only if $[\![\sigma]\!]_\nu \in \mathcal{T}$ for every pair of type σ and type environment ν.

Definition (Validity in a Kripke Model). Let $\mathcal{K} = (\mathcal{W}, \mathfrak{M}, \mathcal{T})$ be a Kripke model. Let Γ and Δ be two finite sequence of statements. Let w be a possible world in $\mathcal{W}$, ξ a term environment in $\mathfrak{M}_w$, and ν a type environment in $\mathcal{T}$ such that $w \in \mathrm{Dom}(\nu(t))$ for every type variable $t \in \mathrm{FTV}(\Gamma) \cup \mathrm{FTV}(\Delta)$. Then, the sequent $\Gamma \rhd \Delta$ is said to be *valid* in $(\mathcal{K}, w, \xi, \nu)$ if and only if either $[\![M]\!]^w_\xi \notin [\![\sigma]\!]_\nu[w]$ for some $M : \sigma$ in Γ, or $[\![N]\!]^w_\xi \in [\![\tau]\!]_\nu[w]$ for some $N : \tau$ in Δ. The sequent $\Gamma \rhd \Delta$ is said to be valid in $\mathcal{K}$ if and only if it is valid in $(\mathcal{K}, w, \xi, \nu)$ for every triple of possible world w in $\mathcal{W}$, term environment ξ in $\mathfrak{M}_w$, and type environment ν in $\mathcal{T}$ such that $w \in \mathrm{Dom}(\nu(t))$ for every type variable $t \in \mathrm{FTV}(\Gamma) \cup \mathrm{FTV}(\Delta)$. Similarly, a statement $M : \sigma$ is said to be valid in $(\mathcal{K}, w, \xi, \nu)$ when so is $\rhd M : \sigma$, and it is said to be valid in $\mathcal{K}$ when so is $\rhd M : \sigma$.

It is proved in in [40] that TLJ+(Simple)+(Eq_β) is complete for Kripke-models.

Theorem 7.3.1 (Completeness of TLJ for Kripke models). *A sequent $\Gamma \rhd M : \sigma$ is derivable in* TLJ + (Simple) + (Eq_β) *if and only if it is valid in all Kripke models.*

We show an application of Kripke models. In Section 2.7 we presented an example showing that $\mathbf{T}_\to$ + ($\mathrm{Axiom}_\mathcal{A}$) + ($\mathrm{Eq}_\beta$) is not complete. Here we repeat the example. We define

$$\mathcal{A} = \{\lambda x.c : p_1 \to p_2,\ d : p_2 \to q,\ d : (p_1 \to p_3) \to q\}$$

with term constants c and d, and type constants p_1, p_2, p_3 and q. Then, $\rhd dc : q$ is valid in any models in which all statements in $\mathcal{A}$ are valid, while it is underivable in $\mathbf{T}_\to$ + ($\mathrm{Axiom}_\mathcal{A}$) + ($\mathrm{Eq}_\beta$). This underivability can be proved with a Kripke model. As presented in Section 2.6, the type interpretation is extended for types with type constants, in a trivial manner. Namely, for each triple of type σ, type environment ν, and constant valuation V, we define the partial domain $[\![\sigma]\!]^V_\nu$ by adding the clause: $[\![a]\!]^V_\nu[w] = V(a)[w]$. It is easily verified that all rules of $\mathbf{T}_\to$ + (Eq_β) are sound in any Kripke model. Therefore, it is enough to construct a Kripke model $\mathcal{K} = (\mathcal{W}, \mathfrak{M}, \mathcal{T})$ and a valuation V in which all axioms are valid and $cd : r$ is not valid. For $\mathcal{W}$ we take the set of two possible worlds w_1 and w_2 with $w_1 \sqsubseteq w_2$. We choose a λ-model $(D, \cdot, [\![-]\!])$ arbitrarily, and we assign it to the possible worlds w_1 and w_2. For $\mathcal{T}$ we take the set of all the subsets of D. For the values of c and d, we define $V(c)$ and $V(d)$ arbitrarily. For the values of type constants we define as follows:

- $\mathrm{Dom}(V(p_1)) = \mathrm{Dom}(V(p_2)) = \mathrm{Dom}(V(p_3)) = \mathrm{Dom}(V(q)) = \{w_1, w_2\}$,
- $V(p_1)[w_1] = V(p_2)[w_1] = V(p_3)[w_1] = V(q)[w_1] = \emptyset$,
- $V(p_1)[w_2] = V(p_2)[w_2] = V(q)[w_2] = D$ and $V(p_3)[w_2] = \emptyset$.

Then, it is easily checked that $[\![p_1 \to p_2]\!]^V_\nu[w] = [\![p_1 \to q]\!]^V_\nu[w] = D$ for $w = w_1, w_2$. Moreover, $[\![p_1 \to p_3]\!]^V_\nu[w] = \emptyset$, so that $[\![(p_1 \to p_3) \to q]\!]^V_\nu[w] = D$. Therefore, all statements in $\mathcal{A}$ are valid in $\mathcal{K}$ at both possible worlds $w = w_1, w_2$. However, $dc : q$ is not valid at w_1 since $V(q)[w_1] = \emptyset$. As a consequent, $dc : q$ is underivable in $\mathbf{T}_\to$ + ($\mathrm{Axiom}_\mathcal{A}$) + ($\mathrm{Eq}_\beta$).

References

[1] F. Alessi and F. Barbanera. Strong conjunction and intersection types. In A. Tarlecki, editor, *Mathematical Foundations of Computer Science 1991*, volume 520 of *Lecture Notes in Computer Science*, pages 64–73, Kazimierz Dolny, Poland, 9–13 Sept. 1991. Springer.

[2] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Intersection and union types: Syntax and semantics. *Inform. Comp.*, 119(2):202–230, June 1995.

[3] H. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabbai, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 117–309. Oxford University Press, 1992.

[4] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. Symbolic Logic*, 48:931–940, 1983.

[5] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, Amsterdam, 1984.

[6] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science*, pages 19–75. Academic Press, London, 1990.

[7] F. Cardone, M. Dezani-Ciancaglini, and U. de'Liguoro. Combining type disciplines. *Ann. Pure Appl. Logic*, 66(3):197–230, 5 Apr. 1994.

[8] A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5, 1940.

[9] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Z. Math. Log. Grund Math.*, 27:45–58, 1981.

[10] H. B. Curry. Functionality in combinatory logic. In *Proc. Nat. Acad. Science USA 20*, pages 584–590, 1934.

[11] H. B. Curry and R. Feys. *Combinatory Logic, Vol. I.* North-Holland, Amsterdam, 1958.

[12] H. B. Curry, J. R. Hindley, and J. P. Seldin. *Combinatory Logic, Vol. II.* North-Holland, Amsterdam, 1972.

[13] N. G. de Bruijn. A survey of the project automath. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 579–606. Academic Press, London, 1980.

[14] M. Dezani-Ciancaglini and M. Coppo. An extension of basic functionality theory for lambda-calculus. *Notre Dame J. Formal Log.*, 21:685–693, 1980.

[15] M. Dezani-Ciancaglini, S. Ghilezan, and B. Venneri. The "relevance" of intersection and union types. *Notre Dame J. Formal Logic*, 38(2):246–269, Spring 1997.

[16] M. Dezani-Ciancaglini and I. Margaria. A characterization of F-complete assignments. *Theoret. Comput. Sci.*, 45(2):121–157, 1986. Fundamental study.

[17] J. H. Gallier. On Girard's "candidats de reductibilité". In P. Odifreddi, editor, *Logic and Computer Science*, pages 123–203. Academic Press, London, 1990.

[18] G. Gentzen. Untersuchungen über das logische Schließen, I, II. *Math. Zeitschr.*, 39:176–210, 405–431, 1934.

[19] J. Y. Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, University of Paris VII, 1972.

[20] R. Hindley. The completeness theorem for typing λ-terms. *Theoret. Comput. Sci.*, 22(1–2):1–17, Jan. 1983.

[21] R. Hindley. Curry's type-rules are complete with respect to the F-semantics too. *Theoret. Comput. Sci.*, 22(1–2):127–133, Jan. 1983.

[22] W. Howard. The formulae-as-types notation of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–501. Academic Press, London, 1980.

[23] D. Leivant. Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68, 1986.

[24] D. MacQueen, G. Plotkin, and R. Sethi. An ideal model for recursive polymorphic types. *Inform. Comp.*, 71(1/2):95–130, Oct./Nov. 1986.

[25] D. MacQueen and R. Sethi. A semantic model of types for applicative languages. In *ACM Symposium on Lisp and Functional Programming*, pages 243–252, 1982.

[26] A. R. Meyer, J. C. Mitchell, E. Moggi, and R. Statman. Empty types in polymorphic lambda calculus. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages*, pages 253–262, Munich, Germany, Jan. 1987.

[27] J. C. Mitchell. Polymorphic type inference and containment. *Inform. Comp.*, 76(2/3):211–249, Feb./Mar. 1988.

[28] J. C. Mitchell and E. Moggi. Kripke-style models for typed lambda calculus. *Ann. Pure Appl. Logic*, 51(1–2):99–124, 1991.

[29] J. C. Mitchell and G. D. Plotkin. Abstract types have existential type. *ACM Trans. Prog. Lang. Sys.*, 10(3):470–502, July 1988.

[30] J. Palsberg and C. Pavlopoulou. From polyvariant flow information to intersection and union types. In *Conference Record of POPL '98: The 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 197–208, San Diego, California, 19–21 Jan. 1998.

[31] B. C. Pierce. Programming with intersection types, union types, and polymorphism. Technical report, CMU-CS-91-106, Carnegie Mellon University, February 1991.

[32] B. C. Pierce. Intersection types and bounded polymorphism. *Mathematical Structures in Computer Science*, 7(2):129–193, Apr. 1997.

[33] G. Plotkin. A semantics for static type inference. *Inform. Comp.*, 109(1/2):256–299, 15 Feb./Mar. 1994.

[34] J. Reynolds. Preliminary design of the programming language Forsythe. Technical report, CMU-CS-88-159, Carnegie Mellon University, June 1988.

[35] J. C. Reynolds. Towards a theory of type structure. In *Mathematical Foundations of Software Development*, volume 146 of *Lecture Notes in Computer Science*, pages 408–426, Berlin, 1972. Springer-Verlag.

[36] H. Yokouchi. F-semantics for type assignment systems. *Theoret. Comput. Sci.*, 129(1):39–77, 20 June 1994. Fundamental Study.

[37] H. Yokouchi. Embedding a second order type system into an intersection type system. *Inform. Comp.*, 117(2):206–220, Mar. 1995.

[38] H. Yokouchi. Completeness of type assignment systems with intersection, union, and type quantifiers (full paper). Available at `http://www.keim.cs.gunma-u.ac.jp/~yokouchi`, 1998.

[39] H. Yokouchi. Completeness of type assignment systems with intersection, union, and type quantifiers. In *Proceedings, Thirteenth Annual IEEE Symposium on Logic in Computer Science*, pages 368–379, Indianapolis, Indiana, 1998. IEEE Computer Society Press.

[40] H. Yokouchi and R. Kashima. Sequent calculi for type assignment and their completeness. Available at `http://www.keim.cs.gunma-u.ac.jp/~yokouchi`, 1997.