INFORMATION AND COMPUTATION 117, 206–220 (1995)

Embedding a Second-Order Type System into an Intersection Type System

HIROFUMI YOKOUCHI

IBM Research, Tokyo Research Laboratory, 1623-14, Shimotsuruma, Yamato-shi, Kanagawa 242, Japan
E-mail: yokouchi@trl.ibm.co.jp

This paper presents the relationship between a second-order type assignment system $\mathbf{T}_\forall$ and an intersection type assignment system $\mathbf{T}_\wedge$. First we define a translation tr from intersection types to second-order types. Then we define a system $\mathbf{T}_{\wedge *}$ obtained from $\mathbf{T}_\wedge$ by restricting the use of the intersection type introduction rule, and show that $\mathbf{T}_{\wedge *}$ and $\mathbf{T}_\forall$ are equivalent in the following senses: (a) if a λ-term M has a type σ in $\mathbf{T}_{\wedge *}$, then M has the type $\mathrm{tr}(\sigma)$ in $\mathbf{T}_\forall$; and conversely, (b) if M has a type τ in $\mathbf{T}_\forall$, then M has a type σ in $\mathbf{T}_{\wedge *}$ such that $\mathrm{tr}(\sigma)$ is equivalent to τ. These two theorems mean that $\mathbf{T}_\forall$ is embedded into $\mathbf{T}_\wedge$. © 1995 Academic Press, Inc.

1. INTRODUCTION

A number of type systems assigning types to type-free λ-terms have been proposed since Curry's classical type assignment system was introduced (Curry and Feys, 1958). This paper deals with two such systems—a second-order (polymorphic) type assignment system $\mathbf{T}_\forall$ and an intersection type assignment system $\mathbf{T}_\wedge$—and presents the relationship between them.

The original version of $\mathbf{T}_\forall$ was introduced as a typed λ-calculus, called the second-order λ-calculus, by Girard (1972) and Reynolds (1974). The system $\mathbf{T}_\forall$ is an extension of Curry's system, formed by introducing a universal quantifier over types so that $\mathbf{T}_\forall$ allows, for example, the following typing whose λ-term has no type in Curry's system:

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

The system $\mathbf{T}_\forall$ is a candidate type system to be used in designing a stronger type structure for programming languages.

The intersection type assignment system $\mathbf{T}_\wedge$ is another extension of Curry's system, formed by introducing a new type forming operator $\wedge$. The original version was introduced by Coppo and Dezani-Ciancaglini (1980) and Coppo *et al.* (1981). A type that has the form $\sigma \wedge \tau$ in interpreted as the type of λ-terms that have both types σ and τ. The system $\mathbf{T}_\wedge$ allows us to assign types to a wide class of λ-terms such as

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

Indeed, the set of λ-terms typable in $\mathbf{T}_\wedge$ exactly coincides with the set of strongly normalizable λ-terms (Pottinger, 1980; Leivant, 1986). Moreover, a variant of $\mathbf{T}_\wedge$ has a universal type ω that can be assigned to any λ-terms. Coppo *et al.* (1981) and Leivant (1986) showed that two classes of normalizable and solvable λ-terms can be characterized by typing in this extended system. For more information, see Cardone and Coppo (1990) and the references therein.

We may hope that these two systems $\mathbf{T}_\forall$ and $\mathbf{T}_\wedge$ are deeply related, as a universal quantifier over types intuitively implies an intersection of infinitely many types. Informally, a second-order type of the form $\forall t.\sigma$ may be regarded as the intersection type $\sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2] \wedge \cdots$ of infinite length, where $\langle \alpha_1, \alpha_2, ... \rangle$ is an enumeration of all types. Indeed, Leivant (1990) has proposed a variant of an intersection type assignment system which allows intersection of infinitely many types. Formally, this interpretation of $\forall t.\sigma$ is a circular argument and not acceptable, because $\sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2] \wedge \cdots$ itself must be a type. However, when $\forall t.\sigma$ is used in the formal system $\mathbf{T}_\forall$, this circularity can be avoided. For example, consider typing $x : \forall t.\sigma \vdash M : \tau$ in $\mathbf{T}_\forall$. The proof tree deriving this typing is finite, and only a finite number of instances of $\forall t.\sigma$ may be used in the proof tree. Therefore $\forall t.\sigma$ may be interpreted as the intersection type that consists of such finite number of instances of $\forall t.\sigma$, provided we interpret the specific typing $x : \forall t.\sigma \vdash M : \tau$. This suggests that for each derivation in $\mathbf{T}_\forall$ we can construct a derivation with the same structure in $\mathbf{T}_\wedge$.

Conversely, one could hope to translate intersection types into second-order types so that typings are preserved. However, this is impossible. Giannini and Ronchi Della Rocca (1988) showed the existence of a λ-term that is typable in $\mathbf{T}_\forall$ but not typable in $\mathbf{T}_\wedge$. Therefore there is no complete correspondence between $\mathbf{T}_\wedge$ and $\mathbf{T}_\forall$. Recall that every strongly normalizable λ-term is typable in $\mathbf{T}_\wedge$ and that every typable λ-term in $\mathbf{T}_\forall$ is strongly normalizable. These facts imply that every typable λ-term in $\mathbf{T}_\forall$ is also typable in $\mathbf{T}_\wedge$. This in turn suggests that typings in $\mathbf{T}_\forall$ may be embedded into some subsystem of $\mathbf{T}_\wedge$.

0890-5401/95 $6.00

In this paper, formalizing these intuitive arguments, we show the relationship between $\mathbf{T}_\wedge$ and $\mathbf{T}_\forall$, and discuss the use of intersection types in investigating the properties of $\mathbf{T}_\forall$. We define a translation tr(—) from intersection types into second-order types and introduce a subsystem $\mathbf{T}_{\wedge}*$ of $\mathbf{T}_\wedge$, which is defined by restricting the use of the intersection type introduction rule. Then we prove the following two main theorems: (a) if a λ-term M has a type σ in $\mathbf{T}_{\wedge}*$, then M has the type $\mathrm{tr}(\sigma)$ in $\mathbf{T}_\forall$; and (b) if M has a type τ in $\mathbf{T}_\forall$, then M has a type σ in $\mathbf{T}_{\wedge}*$ such that $\mathrm{tr}(\sigma)$ is equivalent to τ. These two theorems mean that $\mathbf{T}_{\wedge}*$ and $\mathbf{T}_\forall$ are essentially equivalent.

We remark that we can define no translation from second-order types to intersection types, which will be shown in Section 3. Instead, Theorem (b) means that a derivation tree of typing in $\mathbf{T}_{\wedge}*$ is constructed from a given derivation tree in $\mathbf{T}_\forall$. A similar situation is found in an early work of Gödel (1958), who showed the consistency of the first-order Peano arithmetic on the basis of his system of primitive recursive functionals. The first-order Peano arithmetic is translated into the corresponding intuitionistic Peano arithmetic, and then translated into Gödel's system. In the second translation, a formula such as $(\forall x)(\exists y)\,\phi$ is translated into $\phi[y := f(x)]$ with a primitive functional f, and quantifiers in the formula are eliminated. This translation is applied not to each formula but to a given proof tree in the intuitionistic Peano arithmetic. Similarly, Theorem (b) implies elimination of the quantifier through the type-as-formula isomorphism. The difference is that Theorem (b) deals with the second-order propositional logic.

Theorem (b) states a complex situation. Both systems $\mathbf{T}_\forall$ and $\mathbf{T}_\wedge$ satisfy the strong normalization theorem: if M is typable, then M is strongly normalizable. The strong normalization theorem for $\mathbf{T}_\forall$ has no proof within the second-order Peano arithmetic, while the same theorem for $\mathbf{T}_\wedge$ does. This fact shows that $\mathbf{T}_\forall$ has a much more complex structure than $\mathbf{T}_\wedge$. Theorem (b) means that $\mathbf{T}_\forall$ with a complex structure can be embedded into $\mathbf{T}_\wedge$ with a simple structure. Moreover, Theorem (b) cannot be proved within the second-order Peano arithmetic, since the strong normalization theorem for $\mathbf{T}_\forall$ can be reduced to the same theorem for $\mathbf{T}_\wedge$ by using Theorem (b).

The structure of this paper is as follows. In Section 2 we define the translation tr(—) and the subsystem $\mathbf{T}_{\wedge}*$, and show that $\mathbf{T}_{\wedge}*$ is embedded into $\mathbf{T}_\forall$. In Section 3, conversely, we show that $\mathbf{T}_\forall$ is embedded into $\mathbf{T}_{\wedge}*$. To prove this, we introduce another typed λ-calculus with intersection types, and show that this system naturally corresponds to the second-order typed λ-calculus. In Section 4 we prove the key lemma that is used without proof in Section 3. In this lemma we construct a derivation tree of $\mathbf{T}_\wedge$ from a given derivation tree of $\mathbf{T}_\forall$. Finally, in Section 5, we offer some remarks on applications of the main theorems.

2. FROM THE INTERSECTION TYPE SYSTEM TO THE SECOND-ORDER TYPE SYSTEM

In this section we offer preliminary definitions for the second-order type system and the intersection type system, and examine the translation from the intersection type system into the second-order type system.

To start with, we define the second-order type system $\mathbf{T}_\forall$.

DEFINITION. We assume that an infinite set of *type variables* is specified. Then we define types of $\mathbf{T}_\forall$, called *$\forall$-types*, as follows.

(1) Every type variable is a $\forall$-type.

(2) If σ and τ are $\forall$-types, then $\sigma \to \tau$ is a $\forall$-type.

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

We abbreviate $\sigma_1 \to (\sigma_2 \to \cdots \to (\sigma_{n-1} \to \sigma_n) \cdots)$ to $\sigma_1 \to \sigma_2 \to \cdots \to \sigma_n$.

We define free type variables and substitution for free type variables in the standard way, and write $\sigma[t_1, ..., t_n := \alpha_1, ..., \alpha_n]$ to express the $\forall$-type obtained from a $\forall$-type σ by substituting $\forall$-types $\alpha_1, ..., \alpha_n$ for distinct type variables $t_1, ..., t_n$ simultaneously. For $\forall$-type σ, we define $\mathrm{FTV}(\sigma)$ as the set of all free type variables in σ. We use $s, t, u, ...$ for type variables. In this paper we define several kinds of types in addition to $\forall$-types and use $\rho, \sigma, \tau, \alpha, \beta, \gamma, ...$ for such types.

We define type-free λ-terms, free variables, and substitution in the standard way. We call variables for λ-terms just *variables* to distinguish them from type variables. We use $x, y, z, ...$ for variables and $M, N, ...$ for λ-terms.

We use $\equiv$ as syntactical equality sign for λ-terms and types. As usual, when two λ-terms M and N are the same except for bound variables, we syntactically identify them and write $M \equiv N$. Similarly, when two $\forall$-types σ and τ are the same except for bound type variables, we syntactically identity them and write $\sigma \equiv \tau$.

DEFINITION (Inference Rules of $\mathbf{T}_\forall$). Let Γ be a sequence $\langle x_1 : \sigma_1, ..., x_n : \sigma_n \rangle$ $(n \geqslant 0)$, where $x_1, ..., x_n$ are distinct variables and $\sigma_1, ..., \sigma_n$ are $\forall$-types. Such a sequence is called a *basis*. We have the rules

$$(var) \qquad \Gamma \vdash_\forall x_i : \sigma_i \quad (1 \leqslant i \leqslant n)$$

$$(\to I) \qquad \frac{\Gamma, x : \sigma \vdash_\forall M : \tau}{\Gamma \vdash_\forall \lambda x.M : \sigma \to \tau}$$

$$(\to E) \qquad \frac{\Gamma \vdash_\forall M : \sigma \to \tau \quad \Gamma \vdash_\forall N : \sigma}{\Gamma \vdash_\forall MN : \tau}$$

$$(\forall I) \qquad \frac{\Gamma \vdash_\forall M : \sigma}{\Gamma \vdash_\forall M : \forall t.\sigma}$$

where t is a type variable not free in Γ, and

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

where α is a $\forall$-type.

We next define the intersection type system $\mathbf{T}_\wedge$.

DEFINITION. We define types of $\mathbf{T}_\wedge$, called $\wedge$*-types*, as follows.

(1) Every type variable is a $\wedge$-type.

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

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

We abbreviate $(\cdots(\sigma_1 \wedge \sigma_2) \wedge \cdots) \wedge \sigma_n$ to $\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_n$.

DEFINITION (Inference Rules of $\mathbf{T}_\wedge$). Let Γ be a basis $\langle x_1: \sigma_1, ..., x_n: \sigma_n \rangle$ such that $\sigma_1, ..., \sigma_n$ are $\wedge$-types. We have the rules

$$(var) \qquad \Gamma \vdash_\wedge x_i: \sigma_i \qquad (1 \leqslant i \leqslant n)$$

$$(\to I) \qquad \frac{\Gamma, x: \sigma \vdash_\wedge M: \tau}{\Gamma \vdash_\wedge \lambda x.M: \sigma \to \tau}$$

$$(\to E) \qquad \frac{\Gamma \vdash_\wedge M: \sigma \to \tau \qquad \Gamma \vdash_\wedge N: \sigma}{\Gamma \vdash_\wedge MN: \tau}$$

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

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

The original version of intersection type system has special type ω and axiom $\vdash M: \omega$, while this paper adopts the system without ω. When we consider the relation between the intersection type system and the second-order type system, the system without ω seems more suitable.

In the rest of this section, we define a translation from $\wedge$-types into $\forall$-types. We explain our plan by example. Consider the following typing in $\mathbf{T}_\wedge$, which may be used for typing $(\lambda x.xx)(\lambda y.y)$:

$$\vdash_\wedge \lambda x.xx: (p \to p) \wedge ((p \to p) \to (p \to p)) \to p \to p.$$

This typing is derived as follows:

$$\frac{\dfrac{\dfrac{x:\alpha \vdash_\wedge x:\alpha}{x:\alpha \vdash_\wedge x:(p \to p) \to (p \to p)} \qquad \dfrac{x:\alpha \vdash_\wedge x:\alpha}{x:\alpha \vdash_\wedge x: p \to p}}{x:\alpha \vdash_\wedge xx: p \to p}}{\vdash_\wedge \lambda x.xx: \alpha \to p \to p}$$

where p is a type variable, and $\alpha \equiv (p \to p) \wedge ((p \to p) \to (p \to p))$. We note that the two components of intersection type α have a common pattern $t \to t$. If we replace α by $\forall t.t \to t$ in the above derivation, then we obtain a derivation valid in $\mathbf{T}_\forall$. In order to generalize this idea, we have to clarify what is the exact meaning of "common pattern." We use a preorder $\sqsubseteq$ called a *containment* relation, which has been introduced by Mitchell (1988).

DEFINITION. We define preorder $\sqsubseteq$ on the set of $\forall$-types as follows: $\sigma \sqsubseteq \tau$ iff $\sigma \equiv \forall s_1 \cdots \forall s_l.\sigma^\circ$ and $\tau \equiv \forall t_1 \cdots \forall t_m.\sigma^\circ[s_1,..., s_l := \alpha_1,..., \alpha_l]$ for some distinct type variables $s_1, ..., s_l$, type variables $t_1, ..., t_m$ not free in σ, and $\forall$-types $\alpha_1, ..., \alpha_l$, and σ°. We define $\sigma \cong \tau$ iff $\sigma \sqsubseteq \tau$ and $\tau \sqsubseteq \sigma$.

We show that for every pair of $\forall$-types there exists a greatest lower bound of them with respect to $\sqsubseteq$, which will be used in defining the translation of $\wedge$-types that have the form of $\sigma \wedge \tau$ into $\forall$-types. In order to prove this property, we introduce another preorder $\leqslant$ related to $\sqsubseteq$ and show that there is a greatest lower bound with respect to $\leqslant$.

DEFINITION. (i) We assume that an infinite set of special kind of variables, called *substitution variables*, is specified. Then we extend the definition of $\forall$-types by adding the following clause, and call the $\forall$-types generated from the extended rules *extended* $\forall$*-types*.

(4) Every substitution variable is an extended $\forall$-type.

Note that t of $\forall t.\sigma$ must be a type variable instead of a substitution variable.

(ii) Define preorder $\leqslant$ on the set of all extended $\forall$-types as follows: $\sigma \leqslant \tau$ iff $\sigma[a_1, ..., a_n := \alpha_1, ..., \alpha_n] \equiv \tau$ for some substitution variables $a_1, ..., a_n$ and extended $\forall$-types $\alpha_1, ..., \alpha_n$. Define $\sigma \sim \tau$ iff $\sigma \leqslant \tau$ and $\tau \leqslant \sigma$.

It is well known that the set of all terms generated from a set of variables and function symbols forms a complete lattice with respect to the preorder of substitution like our $\leqslant$. See Plotkin (1970), Reynolds (1970), Huet (1976), and Lassez *et al.* (1986). Note that the preorder $\leqslant$ is reversed in these literatures. We show that the set of extended $\forall$-types forms a lattice as well as the term algebra. We provide an algorithm computing the greatest lower bound, using a similar method introduced by Huet (1976). See also Lassez *et al.* (1986), in which the algorithm computing the greatest lower bound on the lattice of term algebra is called anti-unification and its properties are investigated extensively.

LEMMA 2.1. *For every pair of extended* $\forall$*-types, there exists a greatest lower bound of them with respect to* $\leqslant$.

To prove this lemma, we define a procedure calculating the greatest lower bound.

DEFINITION. (i) Let P be a finite set of type variables. We define binary predicate $\text{check}_P(\cdots, -)$ on the set of extended $\forall$-types. For each pair of extended $\forall$-types σ and

τ, if $P \cap \mathrm{FTV}(\sigma) = \varnothing = P \cap \mathrm{FTV}(\tau)$, then $\mathrm{check}_P(\sigma, \tau)$ is defined as **true**. Otherwise, $\mathrm{check}_P(\sigma, \tau)$ is defined as follows:

$$\mathrm{check}_P(\sigma,\tau) = \begin{cases} \textbf{true} & \\ \quad \text{if} \quad \sigma \equiv \tau \equiv t \quad \text{(type variable)}, & \\ \mathrm{check}_P(\sigma_1, \tau_1) \quad \text{and} \quad \mathrm{check}_P(\sigma_2, \tau_2) & \\ \quad \text{if} \quad \sigma \equiv \sigma_1 \to \sigma_2 \quad \text{and} \quad \tau \equiv \tau_1 \to \tau_2, & \\ \mathrm{check}_{P \cup \{t\}}(\sigma_1, \tau_1) & \\ \quad \text{if} \quad \sigma \equiv \forall t.\sigma_1 \quad \text{and} \quad \tau \equiv \forall t.\tau_1, & \\ \textbf{false} & \\ \quad \text{otherwise.} & \end{cases}$$

(ii) Let ESTYPE/$\equiv$ be the set of all equivalence classes on extended $\forall$-types with respect to $\equiv$, and let STVAR be the set of all substitution variables. Let θ be an injective mapping from (ESTYPE/$\equiv$) $\times$ (ESTYPE/$\equiv$) into STVAR. For each pair of extended $\forall$-types σ and τ we define the extended $\forall$-type $\mathrm{cq}_\theta(\sigma, \tau)$,

$$\mathrm{cq}_\theta(\sigma,\tau) \equiv \begin{cases} t & \\ \quad \text{if} \quad \sigma \equiv \tau \equiv t \quad \text{(type variable)}, & \\ \mathrm{cq}_\theta(\sigma_1, \tau_1) \to \mathrm{cq}_\theta(\sigma_2, \tau_2) & \\ \quad \text{if} \quad \sigma \equiv \sigma_1 \to \sigma_2 \quad \text{and} \quad \tau \equiv \tau_1 \to \tau_2, & \\ \forall t.\mathrm{cq}_\theta(\sigma_1, \tau_1) & \\ \quad \text{if} \quad \sigma \equiv \forall t.\sigma_1, \quad \tau \equiv \forall t.\tau_1, & \\ \quad \text{and} \quad \mathrm{check}_{\{t\}}(\sigma_1, \tau_1) \text{ is } \textbf{true}, & \\ \theta([\sigma]_\equiv, [\tau]_\equiv) & \\ \quad \text{otherwise,} & \end{cases}$$

where $[\sigma]_\equiv$ and $[\tau]_\equiv$ are the equivalence classes containing σ and τ with respect to $\equiv$, respectively.

Note that $\mathrm{cq}_\theta(\sigma, \tau)$ depends on choice of θ, but $\mathrm{cq}_\theta(\sigma, \tau)$ for any θ is all equivalent up to renaming for substitution variables. More precisely, $\mathrm{cq}_\theta(\sigma, \tau) \sim \mathrm{cq}_{\theta'}(\sigma, \tau)$ holds. Therefore we omit the subscript θ of $\mathrm{cq}_\theta(\sigma, \tau)$ whenever no confusion occurs.

The following sublemma explains the role of the predicate $\mathrm{check}_{\{\ \}}(-, -)$.

Sublemma (a). *If* $\forall t_1 \cdots \forall t_n.\rho \leqslant \forall t_1 \cdots \forall t_n.\sigma$ *and* $\forall t_1 \cdots \forall t_n.\rho \leqslant \forall t_1 \cdots \forall t_n.\tau$, *then* $\mathrm{check}_{\{t_1, \ldots, t_n\}}(\sigma, \tau)$ *is* **true**.

Proof. Straightforward. ▮

We now show that $\mathrm{cq}(\sigma, \tau)$ is a greatest lower bound of σ and τ with respect to $\leqslant$.

Sublemma (b). $\mathrm{cq}(\sigma, \tau) \leqslant \sigma$ *and* $\mathrm{cq}(\sigma, \tau) \leqslant \tau$.

Proof. Straightforward. ▮

Sublemma (c). *If* $\rho \leqslant \sigma$ *and* $\rho \leqslant \tau$, *then* $\rho \leqslant \mathrm{cq}(\sigma, \tau)$.

Proof. Let S and T be substitutions for substitution variables $a_1, \ldots, a_n$. Define the substitution U as follows:

$$U = [a_1, \ldots, a_n := \mathrm{cq}(a_1 S, a_1 T), \ldots, \mathrm{cq}(a_n S, a_n T)].$$

It is proved that $\mathrm{cq}(\rho S, \rho T) \equiv \rho U$ is satisfied for every extended $\forall$-type ρ such that the substitution variables in ρ are all contained in $\{a_1, \ldots, a_n\}$, from which the sublemma follows. This is easily proved by induction on the structure of ρ. In the proof, Sublemma (a) may be used. ▮

This completes the proof of Lemma 2.1. Furthermore, we can prove that the set of all equivalence classes of extended $\forall$-types with respect to $\sim$ forms a complete lattice if we add the greatest element. In this paper, however, we do not need this fact.

The existence of greatest lower bound for $\leqslant$ implies the same result for $\sqsubseteq$.

Corollary 2.2. *Every pair of $\forall$-types has a greatest lower bound with respect to* $\sqsubseteq$.

Proof. Let $\sigma \equiv \forall s_1 \cdots \forall s_l.\sigma^\circ$ and $\tau \equiv \forall t_1 \cdots \forall t_m.\tau^\circ$ be two $\forall$-types, where neither σ° nor τ° has the form of $\forall u'.\rho'$. Let $a_1, \ldots, a_l, b_1, \ldots, b_m$ be distinct substitution variables. By Lemma 2.1 we get a greatest lower bound ρ of $\sigma[s_1, \ldots, s_l := a_1, \ldots, a_l]$ and $[t_1, \ldots, t_m := b_1, \ldots, b_m]$ with respect to $\leqslant$. Let $c_1, \ldots, c_n$ be all the substitution variables in ρ. Then, $\forall u_1 \cdots \forall u_n.\rho[c_1, \ldots, c_n := u_1, \ldots, u_n]$ is a greatest lower bound of σ and τ with respect to $\sqsubseteq$, where $u_1, \ldots, u_n$ are distinct type variables. ▮

In general, an equivalence class with respect to $\cong$ contains more than one element. For example, $\forall t_1 \forall t_2 \forall t_3.t_1 \to t_2 \to t_1$, $\forall t_1 \forall t_2.t_1 \to t_2 \to t_1$, and $\forall t_2 \forall t_1.t_1 \to t_2 \to t_1$ are all contained in the same equivalence class. Of them we choose a particular representative such that it has no redundant $\forall$ such as $\forall t_3$ in the above example and that, if it has the form $\forall s \forall t.\sigma$, then s occurs before t in σ. Of the three examples above, only $\forall t_1 \forall t_2.t_1 \to t_2 \to t_1$ satisfies these two conditions. Extending this idea, we introduce an operator reg(–) on $\forall$-types. The operator reg(–) removes redundant $\forall$'s and rearranges bound type variables so that the resulting $\forall$-type satisfies the above conditions. Note that $\mathrm{reg}(\sigma) \cong \sigma$ does not necessarily hold.

Definition. For each $\forall$-type σ we define the regular $\forall$-type $\mathrm{reg}(\sigma)$ as follows.

(1) $\mathrm{reg}(t) \equiv t$.

(2) $\mathrm{reg}(\sigma_1 \to \sigma_2) \equiv \mathrm{reg}(\sigma_1) \to \mathrm{reg}(\sigma_2)$.

(3) Let $\sigma \equiv \forall t_1 \cdots \forall t_n.\sigma^\circ$, where σ° is not of the form $\forall s.\tau$. Define the sequence $\langle s_1, \ldots, s_l \rangle$ of distinct type variables as follows:

- $\{s_1, \ldots, s_l\} = \{t_1, \ldots, t_n\} \cap \mathrm{FTV}(\sigma^\circ)$;
- if $1 \leqslant i \leqslant j \leqslant l$, then the first occurrence of s_i in σ° is to the left of the first occurrence of s_j in σ°.

Note that such a sequence is uniquely determined. Thus, we define

$$\mathrm{reg}(\forall t_1 \cdots \forall t_n.\sigma^\circ) \equiv \forall s_1 \cdots \forall s_l.\, \mathrm{reg}(\sigma^\circ).$$

For example, $\mathrm{reg}(\forall t_1 \forall t_2 \forall t_3.t_1 \to t_2 \to t_1) \equiv \forall t_1 \forall t_2.t_1 \to t_2 \to t_1$.

The following lemma shows basic properties of the operation reg(—).

LEMMA 2.3. *Let σ and τ be $\forall$-types.*

(i) $\mathrm{reg}(\mathrm{reg}(\sigma)) \equiv \mathrm{reg}(\sigma)$.

(ii) $\mathrm{reg}(\forall t.\sigma) \equiv \mathrm{reg}(\forall t.\mathrm{reg}(\sigma))$.

(iii) *If* $\sigma \cong \tau$, *then* $\mathrm{reg}(\sigma) \equiv \mathrm{reg}(\tau)$.

(iv) $\mathrm{reg}(\sigma[t := \tau]) \equiv \mathrm{reg}(\sigma)[t := \mathrm{reg}(\tau)]$.

Proof. Straightforward. ∎

The operation reg(—) plays a role of regularizing $\forall$-types. From Lemma 2.3, for example, it follows that $x_1: \sigma_1, ..., x_n: \sigma_n \vdash_\forall M: \tau$ implies $x_1: \mathrm{reg}(\sigma_1), ..., x_n: \mathrm{reg}(\sigma_n) \vdash_\forall M: \mathrm{reg}(\tau)$.

Now we are ready to define the translation from $\wedge$-types to $\forall$-types.

DEFINITION. For each $\wedge$-type σ, we define the $\forall$-type $\mathrm{tr}(\sigma)$ as follows:

$$\mathrm{tr}(\sigma) \equiv \begin{cases} t & \text{if} \quad \sigma \equiv t \text{ (type variable)}, \\ \mathrm{tr}(\sigma_1) \to \mathrm{tr}(\sigma_2) & \text{if} \quad \sigma \equiv \sigma_1 \to \sigma_2, \\ \mathrm{reg}(\mathrm{tr}(\sigma_1) \sqcap \mathrm{tr}(\sigma_2)) & \text{if} \quad \sigma \equiv \sigma_1 \wedge \sigma_2, \end{cases}$$

where $\sqcap$ means the operation taking a greatest lower bound with respect to $\sqsubseteq$.

The following lemma shows the basic properties of tr(—) and $\sqsubseteq$.

LEMMA 2.4. (i) $\mathrm{reg}(\mathrm{tr}(\sigma)) \equiv \mathrm{tr}(\sigma)$.

(ii) $\mathrm{tr}(\sigma \wedge \tau) \cong \mathrm{tr}(\sigma) \sqcap \mathrm{tr}(\tau)$.

(iii) $\mathrm{tr}(\sigma \wedge \tau) \sqsubseteq \mathrm{tr}(\sigma)$ *and* $\mathrm{tr}(\sigma \wedge \tau) \sqsubseteq \mathrm{tr}(\tau)$.

(iv) *If* $\mathrm{tr}(\sigma) \equiv \mathrm{tr}(\tau)$, *then* $\mathrm{tr}(\sigma \wedge \tau) \equiv \mathrm{tr}(\sigma)$.

(v) *If* $\mathrm{tr}(\sigma)[t := \alpha] \equiv \mathrm{tr}(\tau)$ *and* $t \notin \mathrm{FTV}(\mathrm{tr}(\tau))$, *then* $\mathrm{tr}(\sigma \wedge \tau) \cong \forall t.\mathrm{tr}(\sigma)$.

(vi) *If* $\Gamma \vdash_\forall M: \sigma$ *and* $\sigma \sqsubseteq \tau$, *then* $\Gamma \vdash_\forall M: \tau$.

Proof. They are all easily proved. ∎

Note that, for example, $\mathrm{tr}(\sigma_1 \wedge \sigma_2 \wedge \sigma_3)$ does not depend on the order of σ_1, σ_2, and σ_3. Indeed, by Lemma 2.4(ii),

$$\begin{aligned} \mathrm{tr}(\sigma_1 \wedge \sigma_2 \wedge \sigma_3) &\cong \mathrm{tr}(\sigma_1) \sqcap \mathrm{tr}(\sigma_2) \sqcap \mathrm{tr}(\sigma_3) \\ &\cong \mathrm{tr}(\sigma_2 \wedge \sigma_3 \wedge \sigma_1), \end{aligned}$$

so that, by Lemma 2.3(iii) and 2.4(i),

$$\mathrm{tr}(\sigma_1 \wedge \sigma_2 \wedge \sigma_3) \equiv \mathrm{tr}(\sigma_2 \wedge \sigma_3 \wedge \sigma_1).$$

Using the translation tr(—), one could hope to show that, if M has a type σ in $\mathbf{T}_\wedge$, then M has the translated type $\mathrm{tr}(\sigma)$ in $\mathbf{T}_\forall$. However, this is not true. It is known that all strongly normalizable λ-terms are typable in $\mathbf{T}_\wedge$ (Pottinger,1980; Leivant,1986), but there is a strongly normalizable λ-term M that has no type in $\mathbf{T}_\forall$ (Giannini and Ronchi Della Rocca, 1988). We define a subsystem $\mathbf{T}_{\wedge *}$ of $\mathbf{T}_\wedge$ so that $\mathbf{T}_{\wedge *}$ is embedded into $\mathbf{T}_\forall$ by tr(—).

We consider what is the essential difference between $\mathbf{T}_\forall$ and $\mathbf{T}_\wedge$. It is related to the rule $(\forall I)$ of $\mathbf{T}_\forall$ and the corresponding rule $(\wedge I)$ of $\mathbf{T}_\wedge$. The rule $(\wedge I)$ states that, if $\Gamma \vdash_\forall M: \sigma$ and t is not free in Γ, then $\Gamma \vdash_\forall M: \forall t.\sigma$. Since t is not free in Γ, we have $\Gamma \vdash_\forall M: \sigma[t := \alpha]$ for every α. As explained in the Introduction, we may regard $\forall t.\sigma$ as the intersection $\sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2] \wedge \cdots$ of infinitely many instances of $\forall t.\sigma$. In this intuitive treatment the rule $(\forall I)$ may be interpreted as follows:

$$\frac{\Gamma \vdash M: \sigma[t := \alpha_1] \qquad \Gamma \vdash M: \sigma[t := \alpha_2] \qquad \cdots}{\Gamma \vdash M: \sigma[t := \alpha_1] \wedge \sigma[t := \alpha_2] \wedge \cdots}$$

Therefore the rule $(\wedge I)$ of $\mathbf{T}_\wedge$ may be a special case of this interpretation of $(\forall I)$. However, there is a big difference between $(\wedge I)$ and $(\forall I)$. In the above interpretation of $(\forall I)$, each typing $\Gamma \vdash M: \sigma[t := \alpha_i]$ must be derived by a derivation tree with the same structure as that of $\Gamma \vdash M: \sigma$. On the other hand, the rule $(\wedge I)$ states that $\Gamma \vdash_\wedge M: \rho$ and $\Gamma \vdash_\wedge M: \tau$ imply $\Gamma \vdash_\wedge M: \rho \wedge \tau$. The two derivation trees for $\Gamma \vdash_\wedge M: \rho$ and $\Gamma \vdash_\wedge M: \tau$ may be constructed independently. This is the essential difference between $\mathbf{T}_\forall$ and $\mathbf{T}_\wedge$. In order to obtain the subsystem $\mathbf{T}_{\wedge *}$ completely corresponding to $\mathbf{T}_\forall$, we need to introduce the uniformity of $(\forall I)$ into $(\wedge I)$.

Notation. Let $\Gamma \equiv \langle x_1: \sigma_1, ..., x_n: \sigma_n \rangle$ and $\Delta \equiv \langle x_1: \tau_1, ..., x_n: \tau_n \rangle$ be bases of $\mathbf{T}_\forall$.

(i) $\Gamma \equiv \Delta$ iff $\sigma_i \equiv \tau_i$ for every i $(1 \leqslant i \leqslant n)$.

(ii) $\mathrm{tr}(\Gamma) \equiv \langle x_1: \mathrm{tr}(\sigma_1), ..., x_n: \mathrm{tr}(\sigma_n) \rangle$.

In addition to $\equiv$ and tr(—), we will define several operations and relations on types. For such binary relations (—)*rel*(—) and unary operations *op*(—), we define $\Gamma(rel)\,\Delta$ and $op(\Gamma)$ similarly.

DEFINITION. We define subsystem $\mathbf{T}_{\wedge *}$ of $\mathbf{T}_\wedge$ by restricting the use of $(\wedge I)$ as follows:

$$(\wedge I)^* \qquad \frac{\Gamma \vdash_{\wedge *} M: \sigma \qquad \Gamma \vdash_{\wedge *} M: \tau}{\Gamma \vdash_{\wedge *} M: \sigma \wedge \tau}$$

provided there exists a typing $\Delta \vdash_{\wedge *} M: \rho$ such that $\mathrm{tr}(\Gamma) \equiv \mathrm{tr}(\Delta)$, $\forall t_1 \cdots \forall t_n.\mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\sigma)$, and $\forall t_1 \cdots \forall t_n.\mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\tau)$ for some type variables $t_1, ..., t_n$ not free in $\mathrm{tr}(\Gamma)$.

THEOREM 2.5. *If* $\Gamma \vdash_{\wedge *} M: \sigma$, *then* $\mathrm{tr}(\Gamma) \vdash_\forall M: \mathrm{tr}(\sigma)$.

Proof. The theorem is proved by induction on the derivation of $\Gamma \vdash_{\wedge *} M: \sigma$. We treat only the essential cases where the rule applied at the last step of the derivation is $(\wedge E)$ or $(\wedge I)$. The other cases are clear.

Case 1: $(\wedge E)$. Suppose $\Gamma \vdash_{\wedge *} M: \sigma \wedge \tau$. Then, by induction hypothesis, $\mathrm{tr}(\Gamma) \vdash_\forall M: \mathrm{tr}(\sigma \wedge \tau)$. By Lemma 2.4(iii), $\mathrm{tr}(\sigma \wedge \tau) \sqsubseteq \mathrm{tr}(\sigma)$, and therefore, by Lemma 2.4(vi), $\mathrm{tr}(\Gamma) \vdash_\forall M: \mathrm{tr}(\sigma)$.

Case 2. $(\wedge I)^*$. Suppose $\sigma \equiv \sigma_1 \wedge \sigma_2$, $\Delta \vdash_{\wedge *} M: \rho$, $\mathrm{tr}(\Delta) \equiv \mathrm{tr}(\Gamma)$, $\forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\sigma_1)$, and $\forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\sigma_2)$, where $t_1, ..., t_n$ are not free in $\mathrm{tr}(\Gamma)$. Then, by the induction hypothesis, $\mathrm{tr}(\Delta) \vdash_\forall M: \mathrm{tr}(\rho)$, and therefore $\mathrm{tr}(\Gamma) \vdash_\forall M: \forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho)$. By Lemma 2.4(ii), $\forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\sigma_1) \sqcap \mathrm{tr}(\sigma_2) \cong \mathrm{tr}(\sigma_1 \wedge \sigma_2)$, so that, by Lemma 2.4(vi), $\mathrm{tr}(\Gamma) \vdash_\forall M: \mathrm{tr}(\sigma_1 \wedge \sigma_2)$. ▌

Finally we show an example to explain the difference between $\mathbf{T}_\wedge$ and $\mathbf{T}_{\wedge *}$. Giannini and Ronchi Della Rocca (1988) proved that

$$(\lambda xy . y(xI)(xK))(\lambda z . zz)$$

is typable in $\mathbf{T}_\wedge$ but not in $\mathbf{T}_\forall$, where $I \equiv \lambda x.x$ and $K \equiv \lambda xy.x$. In $\mathbf{T}_\wedge$, we can derive the typings:

$$\vdash_\wedge \lambda xy \colon y(xI)(xK) \colon (\alpha \wedge \beta) \to (\alpha' \to \beta' \to \gamma) \to \gamma,$$

$$\vdash_\wedge \lambda z . zz \colon \alpha,$$

$$\vdash_\wedge \lambda z . zz \colon \beta,$$

where

$$\alpha' \equiv p \to p,$$

$$\alpha \equiv (p \to p) \wedge ((p \to p) \to \alpha') \to \alpha',$$

$$\beta' \equiv (q \to q \to q) \to (q \to q \to q),$$

$$\beta \equiv (q \to q \to q) \wedge ((q \to q \to q) \to \beta') \to \beta'.$$

These three typings are derived in $\mathbf{T}_{\wedge *}$ as well as in $\mathbf{T}_\wedge$, but $\vdash_{\wedge *} \lambda z . zz \colon \alpha \wedge \beta$ is not derived in $\mathbf{T}_{\wedge *}$. Suppose $\vdash_{\wedge *} \lambda z . zz \colon \alpha \wedge \beta$. Then, by Theorem 2.5, $\vdash_{\forall *} \lambda z . zz \colon \mathrm{tr}(\alpha \wedge \beta)$. However, this is impossible since

$$\begin{aligned} \mathrm{tr}(\alpha \wedge \beta) &\equiv \mathrm{reg}(\mathrm{tr}(\alpha) \sqcap \mathrm{tr}(\beta)) \\ &\equiv \mathrm{reg}(((\forall s . s \to s) \to \alpha') \sqcap ((\forall t . t \to t \to t) \to \beta')) \\ &\equiv \forall u_1 \forall u_2 . u_1 \to u_2 \to u_2. \end{aligned}$$

3. FROM THE SECOND-ORDER TYPE SYSTEM TO THE INTERSECTION TYPE SYSTEM

In Section 2 we have shown that the subsystem $\mathbf{T}_{\wedge *}$ of $\mathbf{T}_\wedge$ is embedded into $\mathbf{T}_\forall$ by the operation tr(—) translating $\wedge$-types to $\forall$-types. In other words, tr(—) preserves typing in $\mathbf{T}_{\wedge *}$ and $\mathbf{T}_\forall$. In this section we consider the converse direction of embedding.

One could hope to define a translation, say $\mathrm{tr}^{-1}(\text{—})$, from $\forall$-types to $\wedge$-types such that $\mathrm{tr}^{-1}(\text{—})$ preserves typing in $\mathbf{T}_\forall$ and $\mathbf{T}_{\wedge *}$. However, this is impossible. For example, for every $n \geq 1$, we have the typing in $\mathbf{T}_\forall$:

$$\vdash_\forall \mathrm{repeat}(n) \colon (\forall t.t) \to (\forall t.t),$$

where

$$\mathrm{repeat}(n) \equiv \lambda x . \underbrace{x \ldots x}_{n\text{-times}}.$$

If such a $\mathrm{tr}^{-1}(\text{—})$ were defined, we would have

$$\vdash_\wedge \mathrm{repeat}(n) \colon \mathrm{tr}^{-1}((\forall t.t) \to (\forall t.t)).$$

Namely, there is a fixed $\wedge$-type that can be assigned to repeat(n) for all $n \geq 1$. However, the following lemma shows that there cannot be such a $\wedge$-type.

LEMMA 3.1. *There is no $\wedge$-type σ such that* $\vdash_\wedge$ repeat(n): *σ for every $n \geq 1$.*

Proof. First we define a class of $\wedge$-types and a system $\mathbf{T}_{\wedge d}$ essentially equivalent to $\mathbf{T}_\wedge$. The $\wedge$-types generated by the following rules are called *reduced* $\wedge$-types, which is suggested in Leivant (1983):

(1) Every type variable is a reduced $\wedge$-type.

(2) If $\sigma_1, ..., \sigma_n$, and τ are reduced $\wedge$-types, then $\sigma_1 \wedge \cdots \wedge \sigma_n \to \tau$ is a reduced $\wedge$-type. The system $\mathbf{T}_{\wedge d}$ is defined by restricting $\wedge$-types appearing on the right-hand side of $\vdash_\wedge$ to reduced $\wedge$-types as follows. Let Γ be a basis $\langle x_1 \colon \sigma_{11} \wedge \cdots \wedge \sigma_{1p(1)}, ..., x_n \colon \sigma_{n1} \wedge \cdots \wedge \sigma_{1p(n)} \rangle$ such that each σ_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq p(i))$ is a reduced $\wedge$-type.

$$(var) \qquad \Gamma \vdash_{\wedge d} x_i \colon \sigma_{ij} \qquad (1 \leq i \leq n,\ 1 \leq j \leq p(i))$$

$$(\to I) \qquad \frac{\Gamma, x \colon \sigma_1 \wedge \cdots \wedge \sigma_m \vdash_{\wedge d} M \colon \tau}{\Gamma \vdash_{\wedge d} \lambda x . M \colon \sigma_1 \wedge \cdots \wedge \sigma_m \to \tau}$$

$$(\to E) \qquad \frac{\left[\begin{array}{c} \Gamma \vdash_{\wedge d} M \colon \sigma_1 \wedge \cdots \wedge \sigma_m \to \tau \quad \Gamma \vdash_{\wedge d} N \colon \sigma_i \\ \text{for every } i \quad (1 \leq i \leq m) \end{array}\right]}{\Gamma \vdash_{\wedge d} MN \colon \tau}$$

Here $\sigma_1, ..., \sigma_m$, and τ are reduced $\wedge$-types.

For each $\wedge$-type σ we define the set red(σ) of reduced $\wedge$-types as follows:

$$\mathrm{red}(\sigma) \equiv \begin{cases} \{t\} \\ \qquad \text{if} \quad \sigma \equiv t \quad \text{(type variable)} \\ \mathrm{red}(\sigma_1) \cup \mathrm{red}(\sigma_2) \\ \qquad \text{if} \quad \sigma \equiv \sigma_1 \wedge \sigma_2, \\ \left\{ \bigwedge \mathrm{red}(\sigma_1) \to \alpha \,\middle|\, \alpha \in \mathrm{red}(\sigma_2) \right\} \\ \qquad \text{if} \quad \sigma \equiv \sigma_1 \to \sigma_2. \end{cases}$$

Here, for each finite set $P=\{\alpha_1, \ldots, \alpha_n\}$ of $\wedge$-types, $\bigwedge P$ represents $\alpha_1 \wedge \cdots \wedge \alpha_n$. We choose the order of $\alpha_1, \ldots, \alpha_n$ arbitrarily and fix it for each P. It is easily proved by induction on derivation in $\mathbf{T}_\wedge$ that, if $x_1:\sigma_1, \ldots, x_n:\sigma_n \vdash_\wedge M:\tau$, then $x_1: \bigwedge \mathrm{red}(\sigma_1), \ldots, x_n: \bigwedge \mathrm{red}(\sigma_n) \vdash_{\wedge d} M:\alpha$ for every $\alpha \in \mathrm{red}(\tau)$.

We now prove the present lemma. Suppose that $\vdash_\wedge \mathrm{repeat}(n):\sigma$ for every $n \geqslant 1$. Then, $\vdash_{\wedge d} \mathrm{repeat}(n):\tau$ for every $\tau \in \mathrm{red}(\sigma)$. The derivation of this typing has the following from:

$$\frac{\left[\begin{array}{ll} x:\gamma \vdash_{\wedge d} x:\alpha_1\to\cdots\to\alpha_{n-1}\to\beta & x:\gamma\vdash_{\wedge d} x:\rho \\ \text{for every } \rho\in\mathrm{red}(\alpha_1) & \end{array}\right]}{x:\gamma\vdash_{\wedge d} xx:\alpha_2\to\cdots\to\alpha_{n-1}\to\beta}$$

$$\ddots$$

$$\frac{x:\gamma\vdash_{\wedge d} x\ldots x:\beta}{\vdash_{\wedge d}\lambda x.x\ldots x:\gamma\to\beta}$$

By definition, $\mathrm{red}(\gamma)$ contains $\alpha_1 \to \cdots \to \alpha_{n-1} \to \beta$. Therefore, $\mathrm{red}(\sigma)$ can be represented as $\{\gamma_1 \to \beta_1, \ldots, \gamma_m \to \beta_m\}$ such that each $\mathrm{red}(\gamma_i)$ $(1 \leqslant i \leqslant m)$ contains a reduced $\wedge$-type of the form $\alpha_1 \to \cdots \to \alpha_{n-1} \to \beta_i$. Since $\vdash_\wedge \mathrm{repeat}(n):\sigma$ is satisfied for every $n \geqslant 1$, each $\mathrm{red}(\gamma_i)$ must contain infinitely many $\wedge$-types. This is a contradiction. ∎

It is well known that every λ-term typable in $\mathbf{T}_\forall$ is strongly normalizable (Girard, 1972; Girard *et al.*, 1989) and that every strongly normalizable λ-term is typable in $\mathbf{T}_\wedge$ (Pottinger, 1980; Leivant, 1986). Therefore, if a λ-term M has a $\forall$-type σ in $\mathbf{T}_\forall$, then M has some $\wedge$-type τ in $\mathbf{T}_\wedge$. The discussion above means that such a τ cannot be generated from σ alone, but τ may be generated from M and σ. In this section, we show that a typing $\Delta \vdash_{\wedge *} M:\tau$ in $\mathbf{T}_{\wedge *}$ is uniformly constructed from a derivation of $\Gamma \vdash_\forall M:\sigma$.

Now we are ready to describe the embedding theorem from $\mathbf{T}_\forall$ into $\mathbf{T}_{\wedge *}$. The proof is deferred to the end of this section.

THEOREM 3.2. *If* $\Gamma \vdash_\forall M:\sigma$, *then* $\Delta \vdash_{\wedge *} M:\tau$ *for some* Δ *and* τ *such that* $\mathrm{tr}(\Delta) \equiv \mathrm{reg}(\Gamma)$ *and* $\mathrm{tr}(\tau) \equiv \mathrm{reg}(\sigma)$.

As explained in Section 1, Theorem 3.2 cannot be proved within second-order Peano arithmetic. This suggests that we need some complex device to prove Theorem 3.2. We may use the strong normalization theorem for $\mathbf{T}_\forall$. We construct a derivation of $\Delta \vdash_{\wedge *} M:\tau$ from a given derivation of $\Gamma \vdash_\forall M:\sigma$ by induction on the maximum length of β-reduction sequences started from M to its normal form. However, a λ-term M of $\mathbf{T}_\forall$ does not completely correspond to the structure of the derivation, because M lacks type information. Therefore we use the second-order λ-calculus $\mathbf{T}_{\lambda 2}$ instead of $\mathbf{T}_\forall$.

DEFINITION (Inference Rules of $\mathbf{T}_{\lambda 2}$). Let Γ be a basis $\langle x_1:\sigma_1, \ldots, x_n:\sigma_n\rangle$ such that $\sigma_1, \ldots, \sigma_n$ are $\forall$-types. We have the rules

$$(var) \qquad \Gamma \vdash_{\lambda 2} x_i:\sigma_i \qquad (1 \leqslant i \leqslant n)$$

$$(\to I) \qquad \frac{\Gamma, x:\sigma \vdash_{\lambda 2} M:\tau}{\Gamma \vdash_{\lambda 2} (\lambda x:\sigma.M):\sigma \to \tau}$$

$$(\to E) \qquad \frac{\Gamma \vdash_{\lambda 2} M:\sigma\to\tau \qquad \Gamma \vdash_{\lambda 2} N:\sigma}{\Gamma \vdash_{\lambda 2} MN:\tau}$$

$$(\forall I) \qquad \frac{\Gamma \vdash_{\lambda 2} M:\sigma}{\Gamma \vdash_{\lambda 2} \Lambda t.M:\forall t.\sigma}$$

where t is a type variable not free in Γ, and

$$(\forall E) \qquad \frac{\Gamma \vdash_{\lambda 2} M:\forall t.\sigma}{\Gamma \vdash_{\lambda 2} M\alpha:\sigma[t:=\alpha]}$$

where α is a $\forall$-type.

As usual, two typed λ-terms M and N in $\mathbf{T}_{\lambda 2}$ are syntactically identified whenever M and N are the same except for bound variables and bound type variables. For each typed λ-term M in $\mathbf{T}_{\lambda 2}$ we define $\mathrm{FTV}(M)$ as the set of all free type variables in M.

The next two lemmas are well-known properties of $\mathbf{T}_{\lambda 2}$. For more information about $\mathbf{T}_{\lambda 2}$, see, e.g., Barendregt (1992) and the references therein.

LEMMA 3.3. (i) *If* $\Gamma \vdash_{\lambda 2} M:\sigma$ *and* $\Gamma \vdash_{\lambda 2} M:\tau$, *then* $\sigma \equiv \tau$.

(ii) *If* $\Gamma \vdash_{\lambda 2} M:\sigma$, *then* $\Gamma[t:=\alpha] \vdash_{\lambda 2} M[t:=\alpha]:\sigma[t:=\alpha]$ *for every* $\forall$*-type* α.

(iii) *If* $\Gamma \vdash_{\lambda 2} M:\sigma$ *and* $M \twoheadrightarrow_\beta N$ *(β-reduction), then* $\Gamma \vdash_{\lambda 2} N:\sigma$.

Proof. (i–ii) Straightforward by induction on derivation.

(iii) When $M \to_\beta N$ (1-step β-reduction), it is easily proved by induction on the derivation of $\Gamma \vdash_{\lambda 2} M:\sigma$. ∎

LEMMA 3.4. *For each typed λ-term M allowed in* $\mathbf{T}_{\lambda 2}$, *let* $\mathrm{erase}(M)$ *be the λ-term obtained from M by deleting all type information.*

(i) *If* $\Gamma \vdash_{\lambda 2} M:\sigma$, *then* $\Gamma \vdash_\forall \mathrm{erase}(M):\sigma$.

(ii) *If* $\Gamma \vdash_\forall N:\sigma$, *then* $\Gamma \vdash_{\lambda 2} M:\sigma$ *for some M such that* $\mathrm{erase}(M) \equiv N$.

Proof. (i)–(ii) Straightforward by induction on derivation. ∎

The following theorem is a well-known fact, which cannot be proved within the second-order Peano arithmetic (Girard, 1972; Girard *et al.*, 1989).

FACT (Strong Normalization Theorem for $\mathbf{T}_{\lambda 2}$). *If $\Gamma \vdash_{\lambda 2} M: \sigma$, then there is no infinite sequence of β-reductions started from M.*

DEFINITION. Suppose $\Gamma \vdash_{\lambda 2} M: \sigma$. Then we define $\mathrm{bd}(M)$ as the least number n such that every sequence of β-reductions started from M has less than n-steps. By the strong normalization theorem, $\mathrm{bd}(M)$ is defined whenever M is typable in $\mathbf{T}_{\lambda 2}$.

The proof of Theorem 3.2 is carried out by constructing a derivation tree in $\mathbf{T}_{\wedge}*$ from a derivation tree of $\Gamma \vdash_{\lambda 2} M: \sigma$ in $\mathbf{T}_{\lambda 2}$ by induction on $\mathrm{bd}(M)$. Note that the derivation tree of $\Gamma \vdash_{\lambda 2} M: \sigma$ is determined by M and Γ. The obtained derivation tree in $\mathbf{T}_{\wedge}*$ is expected to have the same structure as the original derivation tree in $\mathbf{T}_{\lambda 2}$. We need to know the structure of derivation tree constructed at each induction step of the proof. In order to formalize the correspondence between derivation trees in $\mathbf{T}_{\wedge}*$ and those in $\mathbf{T}_{\lambda 2}$, we introduce another system $\mathbf{T}_{\lambda\wedge}$, which is a mixed system of $\mathbf{T}_{\lambda 2}$ and $\mathbf{T}_{\wedge}*$.

DEFINITION. We define types σ, called *$\lambda\wedge$-types*, of $\mathbf{T}_{\lambda\wedge}$ and the $\forall$-type $\mathrm{tr}^*(\sigma)$ attached to σ, simultaneously, as follows.

(1) Every type variable t is a $\lambda\wedge$-type, and we define $\mathrm{tr}^*(t) \equiv t$.

(2) Let $\sigma_1, ..., \sigma_n$ $(n \geqslant 1)$, and τ be $\lambda\wedge$-types. If $\mathrm{tr}^*(\sigma_1) \equiv \cdots \equiv \mathrm{tr}^*(\sigma_n)$, then $[\sigma_1, ..., \sigma_n] \rightarrow \tau$ is a $\lambda\wedge$-type, and we define

$$\mathrm{tr}^*([\sigma_1, ..., \sigma_n] \rightarrow \tau) \equiv \mathrm{tr}^*(\sigma_1) \rightarrow \mathrm{tr}^*(\tau).$$

Such an expression $[\sigma_1, ..., \sigma_n]$ is called a *$\lambda\wedge$-l-type*.

(3) Let σ and τ be $\lambda\wedge$-types, and let t be a type variable. If $t \notin \mathrm{FTV}(\mathrm{tr}^*(\tau))$ and $\mathrm{tr}^*(\sigma)[t := \alpha] \equiv \mathrm{tr}^*(\tau)$ for some $\forall$-type α, then $\langle t, \sigma, \tau \rangle$ is a $\lambda\wedge$-type, and we define

$$\mathrm{tr}^*(\langle t, \sigma, \tau \rangle) \equiv \forall t.\mathrm{tr}^*(\sigma).$$

Note that α is uniquely determined when $\mathrm{tr}^*(\sigma) \not\equiv \mathrm{tr}^*(\tau)$; and $t \notin \mathrm{FTV}(\mathrm{tr}^*(\sigma))$ when $\mathrm{tr}^*(\sigma) \equiv \mathrm{tr}^*(\tau)$.

DEFINITION. Let Γ be a sequence $\langle x_1: [\sigma_{11}, ..., \sigma_{1p(1)}], ..., x_n: [\sigma_{n1}, ..., \sigma_{np(n)}] \rangle$ $(n \geqslant 0)$, where $x_1, ..., x_n$ are distinct variables, and each $[\sigma_{i1}, ..., \sigma_{ip(i)}]$ $(1 \leqslant i \leqslant n)$ is a $\lambda\wedge$-l-type. We call such a Γ a *basis* of $\mathbf{T}_{\lambda\wedge}$, and define

$$\mathrm{tr}^*(\Gamma) \equiv \langle x_1: \mathrm{tr}^*(\sigma_{11}), ..., x_n: \mathrm{tr}^*(\sigma_{n1}) \rangle.$$

Note that $\mathrm{tr}^*(\sigma_{i1}) \equiv \cdots \equiv \mathrm{tr}^*(\sigma_{ip(i)})$ $(1 \leqslant i \leqslant n)$.

DEFINITION (Inference Rules of $\mathbf{T}_{\lambda\wedge}$). Let $\Gamma \equiv \langle x_1: [\sigma_{11}, ..., \sigma_{1p(1)}], ..., x_n: [\sigma_{n1}, ..., \sigma_{np(n)}] \rangle$ be a basis of $\mathbf{T}_{\lambda\wedge}$. We have the rules

$$(var) \qquad \Gamma \vdash_{\lambda 2} x_i: \sigma_{ij} \qquad (1 \leqslant i \leqslant n, 1 \leqslant j \leqslant p(i))$$

$$(\rightarrow I) \qquad \frac{\Gamma, x: [\sigma_1, ..., \sigma_m] \vdash_{\lambda\wedge} M: \tau}{\Gamma \vdash_{\lambda\wedge} (\lambda x: \rho.M): [\sigma_1, ..., \sigma_m] \rightarrow \tau}$$

where $\rho \equiv \mathrm{tr}^*(\sigma_1)$;

$$(\rightarrow E) \qquad \frac{\left[\begin{array}{ll} \Gamma \vdash_{\lambda\wedge} M: [\sigma_1, ..., \sigma_m] \rightarrow \tau & \Gamma \vdash_{\lambda\wedge} N: \sigma_i \\ \text{for every } i \quad (1 \leqslant i \leqslant m) & \end{array}\right]}{\Gamma \vdash_{\lambda\wedge} MN: \tau}$$

$$(\langle\ \rangle I) \qquad \frac{\Gamma \vdash_{\lambda\wedge} M: \sigma \quad \Gamma \vdash_{\lambda\wedge} M[t := \alpha]: \tau}{\Gamma \vdash_{\lambda\wedge} \Lambda t.M: \langle t, \sigma, \tau \rangle}$$

where t is a type variable not free in $\mathrm{tr}^*(\Gamma)$ or $\mathrm{tr}^*(\tau)$;

$$(\langle\ \rangle E) \qquad \frac{\Gamma \vdash_{\lambda\wedge} M: \langle t, \sigma, \tau \rangle}{\Gamma \vdash_{\lambda\wedge} M\alpha: \tau}$$

where α is a $\forall$-type such that $\mathrm{tr}^*(\sigma)[t := \alpha] \equiv \mathrm{tr}^*(\tau)$.

The next lemma corresponding to Theorem 2.5 shows that $\mathrm{tr}^*(\text{—})$ preserves typing in $\mathbf{T}_{\lambda\wedge}$ and $\mathbf{T}_{\lambda 2}$. Moreover, it ensures that the above inference rules of $\mathbf{T}_{\lambda\wedge}$ are well-defined.

LEMMA 3.5. *If $\Gamma \vdash_{\lambda\wedge} M: \sigma$, then σ is a $\lambda\wedge$-type and $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} M: \mathrm{tr}^*(\sigma)$.*

Proof. The lemma is proved by induction on the derivation of $\Gamma \vdash_{\lambda\wedge} M: \sigma$. We treat only the essential case where the last step of the derivation is $(\langle\ \rangle I)$. The other cases are clear. Suppose $\Gamma \vdash_{\lambda\wedge} M: \sigma$ and $\Gamma \vdash_{\lambda\wedge} M[t := \alpha]: \tau$, where t is not free in $\mathrm{tr}^*(\Gamma)$ or $\mathrm{tr}^*(\tau)$. Then, by the induction hypothesis, σ and τ are $\lambda\wedge$-types, and we have two typings $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} M: \mathrm{tr}^*(\sigma)$ and $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} M[t := \alpha]: \mathrm{tr}^*(\tau)$. Since t is not free in $\mathrm{tr}^*(\Gamma)$, by Lemma 3.3(ii) we have $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} M[t := \alpha]: \mathrm{tr}^*(\sigma)[t := \alpha]$. By Lemma 3.3(i), $\mathrm{tr}^*(\sigma)[t := \alpha] \equiv \mathrm{tr}^*(\tau)$, and therefore $\langle t, \sigma, \tau \rangle$ is really a $\lambda\wedge$-type. Moreover, applying $(\forall I)$, we obtain $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} \Lambda t.M: \forall t.\mathrm{tr}^*(\sigma)$. ∎

The following lemma shows that $\mathbf{T}_{\lambda 2}$ is embedded into $\mathbf{T}_{\lambda\wedge}$.

LEMMA 3.6. *If $\Gamma \vdash_{\lambda 2} M: \sigma$, then $\Delta \vdash_{\lambda\wedge} M: \tau$ such that $\mathrm{tr}^*(\Delta) \equiv \Gamma$ and $\mathrm{tr}^*(\tau) \equiv \sigma$.*

The proof will be provided in the next section, where a derivation tree in $\mathbf{T}_{\lambda\wedge}$ is constructed from the given derivation tree of $\Gamma \vdash_{\lambda 2} M: \sigma$ by induction on $\mathrm{bd}(M)$. This lemma is the key to this paper. We accept Lemma 3.6 and continue discussion.

A $\lambda\wedge$-type σ means a $\wedge$-type written in a special format, and $\mathrm{tr}^*(\text{—})$ for $\lambda\wedge$-types is closely related to $\mathrm{tr}(\text{—})$ for $\wedge$-types. We define the translation from $\lambda\wedge$-types to $\wedge$-types and examine the relation between $\mathbf{T}_{\lambda\wedge}$ and $\mathbf{T}_{\wedge}*$.

DEFINITION. (i) For each $\lambda\wedge$-type, we define the $\wedge$-type $\mathrm{rep}(\sigma)$ as follows:

$$\mathrm{rep}(\sigma) \equiv \begin{cases} t & \\ \quad \text{if} \quad \sigma \equiv t \quad \text{(type variable)}, \\ \mathrm{rep}(\sigma_1) \wedge \cdots \wedge \mathrm{rep}(\sigma_n) \to \mathrm{rep}(\tau) \\ \quad \text{if} \quad \sigma \equiv [\sigma_1, ..., \sigma_n] \to \tau, \\ \mathrm{rep}(\sigma_1) \wedge \mathrm{rep}(\sigma_2) \\ \quad \text{if} \quad \sigma \equiv \langle t, \sigma_1, \sigma_2 \rangle. \end{cases}$$

(ii) For each basis $\Gamma \equiv \langle x_1: [\sigma_{11}, ..., \sigma_{1p(1)}], ..., x_n: [\sigma_{n1}, ..., \sigma_{np(n)}] \rangle$ of $\mathbf{T}_{\lambda\wedge}$, we define the basis $\mathrm{rep}(\Gamma)$ of $\mathbf{T}_\wedge$ as follows:

$$\mathrm{rep}(\Gamma) \equiv \langle x_1: \mathrm{rep}(\sigma_{11}) \wedge \cdots \wedge \mathrm{rep}(\sigma_{1p(1)}), ..., \\ x_n: \mathrm{rep}(\sigma_{n1}) \wedge \cdots \wedge \mathrm{rep}(\sigma_{np(n)}) \rangle.$$

LEMMA 3.7. $\mathrm{reg}(\mathrm{tr}^*(\sigma)) \equiv \mathrm{tr}(\mathrm{rep}(\sigma))$ *for every* $\lambda\wedge$-*type* σ.

Proof. The proof is by induction on the structure of σ. We treat only the essential case where $\sigma \equiv \langle t, \sigma_1, \sigma_2 \rangle$. By definition, $\mathrm{tr}^*(\sigma_1)[t := \alpha] \equiv \mathrm{tr}^*(\sigma_2)$ for some $\forall$-type α. By Lemma 2.3(iv), $\mathrm{reg}(\mathrm{tr}^*(\sigma_1))[t := \mathrm{reg}(\alpha)] \equiv \mathrm{reg}(\mathrm{tr}^*(\sigma_2))$, and therefore, by the induction hypothesis, $\mathrm{tr}(\mathrm{rep}(\sigma_1))[t := \mathrm{reg}(\alpha)] \equiv \mathrm{tr}(\mathrm{rep}(\sigma_2))$. By definition, $t \notin \mathrm{FTV}(\mathrm{tr}^*(\sigma_2)) = \mathrm{FTV}(\mathrm{tr}(\mathrm{rep}(\sigma_2)))$, so that, by Lemma 2.4(v), $\mathrm{tr}(\mathrm{rep}(\sigma_1) \wedge \mathrm{rep}(\sigma_2)) \cong \forall t.\mathrm{tr}(\mathrm{rep}(\sigma_1))$. Finally, using Lemmas 2.3(ii) and 2.4(i), we conclude that $\mathrm{reg}(\mathrm{tr}^*(\sigma)) \equiv \mathrm{tr}(\mathrm{rep}(\sigma))$. ∎

LEMMA 3.8. *If* $\Gamma \vdash_{\lambda\wedge} M: \sigma$, *then* $\mathrm{rep}(\Gamma) \vdash_{\wedge^*} \mathrm{erase}(M): \mathrm{rep}(\sigma)$.

Proof. The lemma is proved by induction on the derivation of $\Gamma \vdash_{\lambda\wedge} M: \sigma$. We treat only the essential case where the last step of the derivation is $(\langle\ \rangle I)$. Suppose $\Gamma \vdash_{\lambda\wedge} M: \sigma$ and $\Gamma \vdash_{\lambda\wedge} M[t := \alpha]: \tau$, where t is not free in $\mathrm{tr}^*(\Gamma)$ or $\mathrm{tr}^*(\tau)$. Then, by the induction hypothesis, we have $\mathrm{rep}(\Gamma) \vdash_{\wedge^*} \mathrm{erase}(M): \mathrm{rep}(\sigma)$ and $\mathrm{rep}(\Gamma) \vdash_{\wedge^*} \mathrm{erase}(M): \mathrm{rep}(\tau)$. Since $\mathrm{tr}^*(\sigma)[t := \alpha] \equiv \mathrm{tr}^*(\tau)$, by Lemmas 3.7 and 2.3 we have $\mathrm{tr}(\mathrm{rep}(\sigma))[t := \mathrm{reg}(\alpha)] \equiv \mathrm{tr}(\mathrm{rep}(\tau))$, and thus $\forall t.\mathrm{tr}(\mathrm{rep}(\sigma)) \sqsubseteq \mathrm{tr}(\mathrm{rep}(\tau))$. Moreover, t is not free in $\mathrm{reg}(\mathrm{tr}^*(\Gamma)) \equiv \mathrm{tr}(\mathrm{rep}(\Gamma))$. Therefore we can apply $(\wedge I)^*$ and obtain $\mathrm{rep}(\Gamma) \vdash_{\wedge^*} \mathrm{erase}(M): \mathrm{rep}(\sigma) \wedge \mathrm{rep}(\tau)$. ∎

Finally, combining the lemmas, we obtain Theorem 3.2.

Proof of Theorem 3.2. Suppose $\Gamma \vdash_\forall M: \sigma$. Then, by Lemmas 3.4(ii) and 3.6, there exist Γ', M', and σ' such that $\Gamma' \vdash_{\lambda\wedge} M': \sigma'$, $\mathrm{erase}(M') \equiv M$, $\mathrm{tr}^*(\Gamma') \equiv \Gamma$, and $\mathrm{tr}^*(\sigma') \equiv \sigma$. Therefore, by Lemma 3.8, $\mathrm{rep}(\Gamma') \vdash_{\wedge^*} M: \mathrm{rep}(\sigma')$. Finally, by Lemma 3.7, $\mathrm{tr}(\mathrm{rep}(\Gamma')) \equiv \mathrm{reg}(\Gamma)$ and $\mathrm{tr}(\mathrm{rep}(\sigma')) \equiv \mathrm{reg}(\sigma)$. ∎

Through the proof of Theorem 3.2 above we notice that we can postulate a stronger condition on $(\wedge I)^*$ of $\mathbf{T}_{\wedge^*}$. Namely, Theorem 3.2 still holds even if $(\wedge I)^*$ is replaced by the following rule:

$$(\wedge I)' \qquad \frac{\Gamma \vdash_{\wedge^*} M: \sigma \qquad \Gamma \vdash_{\wedge^*} M: \tau}{\Gamma \vdash_{\wedge^*} M: \sigma \wedge \tau}$$

provided $\mathrm{tr}(\sigma)[t := \alpha] \equiv \mathrm{tr}(\tau)$ for some $\forall$-type α and some type variable t not free in $\mathrm{tr}(\Gamma)$. Obviously, the condition on $(\wedge I)'$ is stronger than the condition on $(\wedge I)^*$, and thus Theorem 2.5 still holds for $(\wedge I)'$.

4. PROOF OF THE MAIN LEMMA

This section is devoted to proving Lemma 3.6. Here we repeat the lemma:

If $\Gamma \vdash_{\lambda 2} M: \sigma$, then $\Delta \vdash_{\lambda\wedge} M: \tau$ for some Δ and τ such that $\mathrm{tr}^*(\Delta) \equiv \Gamma$ and $\mathrm{tr}^*(\tau) \equiv \sigma$.

To prove this lemma, we need to construct Δ and τ from the derivation of $\Gamma \vdash_{\lambda 2} M: \sigma$. However, when M is either a variable x, an application $M_1 M_2$, or a type application $M_1 \alpha$, we can strengthen the lemma. In this case, for any $\lambda\wedge$-type ρ such that $\mathrm{tr}^*(\rho) \equiv \sigma$, there exists Δ such that $\Delta \vdash_{\lambda\wedge} M: \rho$ and $\mathrm{tr}^*(\Delta) \equiv \Gamma$. In other words, we can choose any τ we like in this special case. This is a key to the proof.

Another key to the proof is the rule $(\langle\ \rangle I)$ of $\mathbf{T}_{\lambda\wedge}$. When a $\lambda\wedge$-type σ is translated to the $\forall$-type $\mathrm{tr}^*(\sigma)$, the type variable t of $\langle t, \rho, \tau \rangle$ in σ plays a role like a bound type variable. However, a bound type variable can be renamed freely in $\mathbf{T}_{\lambda 2}$, while such t of $\langle t, \rho, \tau \rangle$ cannot be renamed in $\mathbf{T}_{\lambda\wedge}$. The two $\lambda\wedge$-types $\langle t, \rho, \tau \rangle$ and $\langle s, \rho[t := s], \tau[t := s] \rangle$ are completely different. Therefore we have to pay special attention to the use of the inference rule $(\langle\ \rangle I)$ in which a $\lambda\wedge$-type of the form $\langle t, \rho, \tau \rangle$ is introduced. We postulate stronger conditions on $(\langle\ \rangle I)$ in this section.

For each specific inference by $(\langle\ \rangle I)$, the type variable t of $\langle t, \sigma, \tau \rangle$ introduced by $(\langle\ \rangle I)$ is called the *eigen type variable* of the inference. In this section we assume that rule $(\langle\ \rangle I)$ of $\mathbf{T}_{\lambda\wedge}$

$$\frac{\Gamma \vdash_{\lambda\wedge} M: \sigma \qquad \Gamma \vdash_{\lambda\wedge} M[t := \alpha]: \tau}{\Gamma \vdash_{\lambda\wedge} \Lambda t.M: \langle t, \sigma, \tau \rangle}$$

can be applied only when the following two conditions are satisfied:

(1) t is not free in $\mathrm{tr}^*(\Gamma)$ or $\mathrm{tr}^*(\tau)$,

(2) t is not used as an eigen type variable in the derivation of $\Gamma \vdash_{\lambda\wedge} M: \sigma$.

This restricted form of $(\langle\ \rangle I)$ is denoted by $(\langle\ \rangle I)^\circ$. Let P be a finite set of type variables. When $\Gamma \vdash_{\lambda\wedge} M: \sigma$ is derived by some derivation tree in which no eigen type variables are contained in P, we write $\Gamma \vdash^P_{\lambda\wedge} M: \sigma$. Note that $\Gamma \vdash^P_{\lambda\wedge} \Lambda t.M: \langle t, \sigma, \tau \rangle$ implies that $\Gamma \vdash^{P \cup \{t\}}_{\lambda\wedge} M: \sigma$ and $t \notin P$. We use P to control the choice of eigen type variables when we construct a derivation of $\Delta \vdash_{\lambda\wedge} M: \tau$ from the given derivation of $\Gamma \vdash_{\lambda 2} M: \sigma$.

On the basis of these remarks above, we prove Lemma 3.6 in the following form.

THEOREM 4.1. *Let P be a finite set of type variables. Suppose $\Gamma \vdash_{\lambda 2} M: \sigma$. Then there exist $\tilde{\Gamma}$ and $\tilde{\sigma}$ such that $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} M: \tilde{\sigma}$, $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$, and $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma$.*

Furthermore, suppose that the normal form of M is either a variable, an application, or a type application. For any $\wedge$-type $\tilde{\tau}$, if $\mathrm{tr}^*(\tilde{\tau}) \equiv \sigma$, then there exists $\tilde{\Gamma}$ such that $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} M: \tilde{\tau}$ and $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$.

First we introduce a technical notation and lemmas for $\mathbf{T}_{\lambda\wedge}$.

Notation. Let

$$\Gamma \equiv \langle x_1: [\sigma_{11}, ..., \sigma_{1p(1)}], ..., x_n: [\sigma_{n1}, ..., \sigma_{np(n)}]\rangle$$

and

$$\Delta \equiv \langle x_1: [\tau_{11}, ..., \tau_{1q(1)}], ..., x_n: [\tau_{n1}, ..., \tau_{nq(n)}]\rangle$$

be bases of $\mathbf{T}_{\lambda\wedge}$. When $\mathrm{tr}^*(\Gamma) \equiv \mathrm{tr}^*(\Delta)$, we define the basis $\Gamma + \Delta$ of $\mathbf{T}_{\lambda\wedge}$ as follows:

$$\Gamma + \Delta \equiv \langle x_1: [\sigma_{11}, ..., \sigma_{1p(1)}, \tau_{11}, ..., \tau_{1q(1)}], ..., x_n: [\sigma_{n1}, ..., \sigma_{np(n)}, \tau_{n1}, ..., \tau_{nq(n)}]\rangle.$$

LEMMA 4.2. *Let P be a finite set of type variables.*

(i) *If $\Gamma, x: [\sigma_1, ..., \sigma_n] \vdash^{P}_{\lambda\wedge} M: \sigma$ and $\mathrm{tr}^*(\rho) \equiv \mathrm{tr}^*(\sigma_1)$, then*

$$\Gamma, x: [\sigma_1, ..., \sigma_i, \rho, \sigma_{i+1}, ..., \sigma_n] \vdash^{P}_{\lambda\wedge} M: \tau \qquad (0 \leqslant i \leqslant n).$$

(ii) *Let Γ and Δ be bases of $\mathbf{T}_{\lambda\wedge}$ such that $\mathrm{tr}^*(\Gamma) \equiv \mathrm{tr}^*(\Delta)$. If $\Gamma \vdash^{P}_{\lambda\wedge} M: \sigma$, then $\Gamma + \Delta \vdash^{P}_{\lambda\wedge} M: \sigma$ and $\Delta + \Gamma \vdash^{P}_{\lambda\wedge} M: \sigma$.*

Proof. Straightforward. ∎

We now start proving Theorem 4.1 by induction on $\mathrm{bd}(M)$ as explained in the last section. The following lemma is the induction basis.

LEMMA 4.3. *When M is in normal form, Theorem 4.1 holds.*

Proof. First note that $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma$ is satisfied whenever $\Gamma \vdash_{\lambda 2} M: \sigma$, $\tilde{\Gamma} \vdash_{\lambda\wedge} M: \tilde{\sigma}$, and $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$. This follows from Lemmas 3.5 and 3.3(i).

The lemma is proved by induction on the structure of M.

Case 1. $M \equiv M_1 M_2$. Since $\Gamma \vdash_{\lambda 2} M: \sigma$, we have $\Gamma \vdash_{\lambda 2} M_1: \rho \to \sigma$ and $\Gamma \vdash_{\lambda 2} M_2: \rho$ for some $\forall$-type ρ. By the induction hypothesis, $\tilde{\Gamma}_2 \vdash^{P}_{\lambda\wedge} M_2: \tilde{\rho}$ for some $\tilde{\Gamma}_2$ and $\tilde{\rho}$ such that $\mathrm{tr}^*(\tilde{\Gamma}_2) \equiv \Gamma$ and $\mathrm{tr}^*(\tilde{\rho}) \equiv \rho$. Note that M_1 is either a variable, an application, or a type application. Since $\mathrm{tr}^*([\tilde{\rho}] \to \tilde{\tau}) \equiv \rho \to \sigma$, by the induction hypothesis we have $\tilde{\Gamma}_1 \vdash^{P}_{\lambda\wedge} M_1: [\tilde{\rho}] \to \tilde{\tau}$ for some $\tilde{\Gamma}_1$ such that $\mathrm{tr}^*(\tilde{\Gamma}_1) \equiv \Gamma$. Therefore, using Lemma 4.2(ii), we obtain $\tilde{\Gamma}_1 + \tilde{\Gamma}_2 \vdash^{P}_{\lambda\wedge} M_1 M_2: \tilde{\tau}$ and $\mathrm{tr}^*(\tilde{\Gamma}_1 + \tilde{\Gamma}_2) \equiv \Gamma$.

Case 2. $M \equiv \Lambda t.M_1$. Assume that t is not free in Γ. Since $\Gamma \vdash_{\lambda 2} M: \sigma$, we have $\Gamma \vdash_{\lambda 2} M_1: \rho$ for some ρ such that $\forall t.\rho \equiv \sigma$. Let $P' = P \cup \{t\}$. Then, by the induction hypothesis, $\tilde{\Gamma}_1 \vdash^{P'}_{\lambda\wedge} M_1: \tilde{\sigma}_1$ for some $\tilde{\Gamma}_1$ and $\tilde{\sigma}_1$ such that $\mathrm{tr}^*(\tilde{\Gamma}_1) \equiv \Gamma$. Let s be a type variable different from t. Then, by Lemma 3.3(ii), $\Gamma \vdash_{\lambda 2} M_1[t := s]: \sigma[t := s]$. By the induction hypothesis, $\tilde{\Gamma}_2 \vdash^{P}_{\lambda\wedge} M_1[t := s]: \tilde{\sigma}_2$ for some $\tilde{\Gamma}_2$ and $\tilde{\sigma}_2$ such that $\mathrm{tr}^*(\Delta_2) \equiv \Gamma$. Let $\tilde{\Gamma} \equiv \tilde{\Gamma}_1 + \tilde{\Gamma}_2$. Then, by Lemma 4.2(ii), $\tilde{\Gamma} \vdash^{P'}_{\lambda\wedge} M_1: \tilde{\sigma}_1$ and $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} M_1[t := s]: \tilde{\sigma}_2$. We may assume $t \notin P$. Therefore, applying $(\langle\ \rangle)^{\circ}$, we have $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} \Lambda t.M_1: \langle t, \tilde{\sigma}_1, \tilde{\sigma}_2\rangle$.

The other cases are similar. ∎

The following lemma is to be used in the proof of Lemma 4.5 below.

LEMMA 4.4. *Let P and Q be finite sets of type variables such that $Q \subseteq P$. Suppose that:*

(1) $\Gamma \vdash^{P}_{\lambda\wedge} M[x := N]: \tau$,

(2) $\Gamma \vdash^{Q}_{\lambda\wedge} N: \sigma$,

(3) $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M: \mathrm{tr}^*(\tau)$,

(4) $\mathrm{FTV}(\mathrm{tr}^*(\sigma)) \subseteq P$.

Then there exists $\lambda\wedge$-types $\sigma_1, ..., \sigma_n$ such that $\Gamma, x: [\sigma_1, ..., \sigma_n] \vdash^{P}_{\lambda\wedge} M: \tau$ and $\Gamma \vdash^{Q}_{\lambda\wedge} N: \sigma_i$ $(1 \leqslant i \leqslant n)$.

Proof. First note that the condition $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M: \mathrm{tr}^*(\tau)$ can be replaced by the weaker condition: $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M: \tau'$ for some $\forall$-type τ'. Indeed, this weaker condition implies that $\tau' \equiv \mathrm{tr}^*(\tau)$ under the rest of the conditions, which is proved as follows. Since $\Gamma \vdash_{\lambda\wedge} N: \sigma$, by Lemma 3.5 we have $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} N: \mathrm{tr}^*(\sigma)$, and therefore, $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} (\lambda x: \mathrm{tr}^*(\sigma).M)N: \tau'$. By Lemma 3.3(iii), $\mathrm{tr}^*(\Gamma) \vdash_{\lambda 2} M[x := N]: \tau'$. Since $\Gamma \vdash_{\lambda\wedge} M[x := N]: \tau$, by Lemmas 3.5 and 3.3(i) we conclude that $\tau' \equiv \mathrm{tr}^*(\tau)$.

The present lemma is proved by induction on the structure of M.

Case 1. $M \equiv M_1 M_2$. Since $\Gamma \vdash^{P}_{\lambda\wedge} M[x := N]: \tau$, we have $\Gamma \vdash^{P}_{\lambda\wedge} M_1[x := N]: [\tau_1, ..., \tau_m] \to \tau$ and $\Gamma \vdash^{P}_{\lambda\wedge} M_2[x := N]: \tau_j$ $(1 \leqslant j \leqslant m)$ for some $\lambda\wedge$-types $\tau_1, ..., \tau_m$. Since $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M: \mathrm{tr}^*(\tau)$, we have $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M_1: \rho \to \mathrm{tr}^*(\tau)$ and $\mathrm{tr}^*(\Gamma), x: \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M_2: \rho$ for some $\forall$-type ρ. Recall the remark at the beginning. Then, by the induction hypothesis, there exist $\lambda\wedge$-types $\sigma_{01}, ..., \sigma_{0p(0)}, ..., \sigma_{m1}, ..., \sigma_{mp(m)}$ such that

$$\Gamma, x: [\sigma_{01}, ..., \sigma_{0p(0)}] \vdash^{P}_{\lambda\wedge} M_1: [\tau_1, ..., \tau_m] \to \tau,$$

$$\Gamma, x: [\sigma_{j1}, ..., \sigma_{jp(j)}] \vdash^{P}_{\lambda\wedge} M_2: \tau_j \quad (1 \leqslant j \leqslant m),$$

and

$$\Gamma \vdash^{Q}_{\lambda\wedge} N : \sigma_{jk} \qquad (0 \leqslant j \leqslant m,\ 1 \leqslant k \leqslant p(j)).$$

By Lemmas 3.5 and 3.3(i), $\mathrm{tr}^*(\sigma_{jk}) \equiv \mathrm{tr}^*(\sigma)$ for every σ_{jk} $(1 \leqslant j \leqslant m,\ 1 \leqslant k \leqslant p(j))$. Therefore, using Lemma 4.2(i), we have $\Gamma, x : [\sigma_{01}, ..., \sigma_{mp(m)}] \vdash^{P}_{\lambda\wedge} M_1 M_2 : \tau$.

Case 2. $M \equiv \Lambda t.M_1$. The derivation of $\Gamma \vdash_{\lambda\wedge} M[x := N] : \tau$ ends with the form

$$\frac{\Gamma \vdash_{\lambda\wedge} M_2 : \tau_1 \qquad \Gamma \vdash_{\lambda\wedge} M_2[s := \alpha] : \tau_2}{\Gamma \vdash_{\lambda\wedge} \Lambda s.M_2 : \langle s, \tau_1, \tau_2 \rangle}$$

where $\Lambda s.M_2 \equiv M[x := N]$ and $\langle s, \tau_1, \tau_2 \rangle \equiv \tau$. Let $P' = P \cup \{s\}$. Then by definition,

$$\Gamma \vdash^{P'}_{\lambda\wedge} M_2 : \tau_1 \qquad \text{and} \qquad \Gamma \vdash^{P}_{\lambda\wedge} M_2[s := \alpha] : \tau_2.$$

Note that $s \notin P$ by definition. In turn, since $\mathrm{tr}^*(\Gamma), x : \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M : \mathrm{tr}^*(\tau)$, we have $\mathrm{tr}^*(\Gamma), x : \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M_1 : \rho$ for some ρ such that $\forall t.\rho \equiv \mathrm{tr}^*(\tau)$. Here we assume that t is not free in $\mathrm{tr}^*(\Gamma)$ or $\mathrm{tr}^*(\sigma)$. By Lemma 3.3(ii),

$$\mathrm{tr}^*(\Gamma), x : \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M_1[t := s] : \rho[t := s]$$

and

$$\mathrm{tr}^*(\Gamma), x : \mathrm{tr}^*(\sigma) \vdash_{\lambda 2} M_1[t := \alpha] : \rho[t := \alpha].$$

We may assume that t is not free in N. Then, $M_2 \equiv M_1[t := s][x := N]$ and $M_2[s := \alpha] \equiv M_1[t := \alpha][x := N]$ since $\Lambda s.M_2 \equiv \Lambda t.M_1[x := N]$. Therefore, by the induction hypothesis, there exist $\lambda\wedge$-types $\sigma_1, ..., \sigma_l, \sigma_{l+1}, ..., \sigma_{l+m}$ such that

$$\Gamma, x : [\sigma_1, ..., \sigma_l] \vdash^{P'}_{\lambda\wedge} M_1[t := s] : \tau_1,$$

$$\Gamma, x : [\sigma_{l+1}, ..., \sigma_{l+m}] \vdash^{P}_{\lambda\wedge} M_1[t := \alpha] : \tau_2,$$

and

$$\Gamma \vdash^{Q}_{\lambda\wedge} N : \sigma_i \qquad (1 \leqslant i \leqslant l+m).$$

By Lemmas 3.5 and 3.3(i), $\mathrm{tr}^*(\sigma_i) \equiv \mathrm{tr}^*(\sigma)$ for every i $(1 \leqslant i \leqslant l+m)$, so that, by Lemma 4.2(i), we have

$$\Gamma, x : [\sigma_1, ..., \sigma_{l+m}] \vdash^{P'}_{\lambda\wedge} M_1[t := s] : \tau_1$$

and

$$\Gamma, x : [\sigma_1, ..., \sigma_{l+m}] \vdash^{P}_{\lambda\wedge} M_1[t := \alpha] : \tau_2.$$

By condition (4) of the present lemma, $s \notin \mathrm{FTV}(\mathrm{tr}^*(\sigma))$, since $s \notin P$. Therefore, we can apply $(\langle\ \rangle I)^{\circ}$ and obtain

$$\Gamma, x : [\sigma_1, ..., \sigma_{l+m}] \vdash^{P}_{\lambda\wedge} \Lambda s.M_1[t := s] : \langle s, \tau_1, \tau_2 \rangle.$$

Here $\Lambda s.M_1[t := s] \equiv \Lambda t.M_1$.

The other cases are similar. ∎

The induction step of the proof of Theorem 4.1 is carried out by the next lemma.

Notation. For each typed λ-term M in $\mathbf{T}_{\lambda 2}$, if M is not in normal form, then we define $\mathrm{left}(M)$ as the typed λ-term obtained from M by replacing the leftmost redex $(\lambda x : \rho.M_1) M_2$ or $(\Lambda t.M_1)\alpha$ by its contractum $M_1[x := M_2]$ or $M_1[t := \alpha]$, respectively.

LEMMA 4.5. *Let M be a typed λ-term not in normal form. Suppose that the first half of Theorem* 4.1 *holds for any M' such that* $\mathrm{bd}(M') < \mathrm{bd}(M)$; *namely, if $\Gamma' \vdash_{\lambda 2} M' : \sigma'$, then for any finite set P' of type variables there exist Γ'' and σ'' such that $\Gamma'' \vdash^{P'}_{\lambda\wedge} M' : \sigma''$, $\mathrm{tr}^*(\Gamma'') \equiv \Gamma'$, and $\mathrm{tr}^*(\sigma'') \equiv \sigma'$. Let P be a finite set of type variables and suppose that*:

(1) $\Gamma \vdash_{\lambda 2} M : \sigma$,

(2) $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} \mathrm{left}(M) : \tilde{\sigma}$,

(3) $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$,

(4) $\mathrm{FTV}(M) \subseteq P$.

Then there exist $\tilde{\Delta}$ and $\tilde{\tau}$ such that $\tilde{\Delta} \vdash^{P}_{\lambda\wedge} M : \tilde{\tau}$, $\mathrm{tr}^*(\tilde{\Delta}) \equiv \Gamma$, and $\mathrm{tr}^*(\tilde{\tau}) \equiv \sigma$.

Furthermore, if M is either a variable, an application, or a type application, then there exists $\tilde{\Delta}$ such that $\tilde{\Delta} \vdash^{P}_{\lambda\wedge} M : \tilde{\sigma}$ and $\mathrm{tr}^*(\tilde{\Delta}) \equiv \Gamma$.

Proof. First note that $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma$ is always satisfied, which is easily proved using Lemmas 3.5, 3.3(iii), and 3.3(i). Note also that, by Lemmas 3.5 and 3.3(i), $\mathrm{tr}^*(\tilde{\tau}) \equiv \sigma$ is satisfied whenever $\tilde{\Delta} \vdash_{\lambda\wedge} M : \tilde{\tau}$ and $\mathrm{tr}^*(\tilde{\Delta}) \equiv \Gamma$.

The lemma is proved by induction on the structure of M.

Case 1. $M \equiv \lambda x : \rho.M_1$. Straightforward.

Case 2. $M \equiv M_1 M_2$. We distinguish three more subcases.

Subcase 2.1. $\mathrm{left}(M) \equiv \mathrm{left}(M_1)\, M_2$. First note that M_1 is in the form of application or type application. Since $\Gamma \vdash_{\lambda 2} M : \sigma$, we have $\Gamma \vdash_{\lambda 2} M_1 : \rho \to \sigma$ and $\Gamma \vdash_{\lambda 2} M_2 : \rho$ for some $\forall$-type ρ. Since $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} \mathrm{left}(M_1)\, M_2 : \tilde{\sigma}$, we have $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} \mathrm{left}(M_1) : [\tilde{\rho}_1, ..., \tilde{\rho}_n] \to \tilde{\sigma}$ and $\tilde{\Gamma} \vdash^{P}_{\lambda\wedge} M_2 : \tilde{\rho}_i$ $(1 \leqslant i \leqslant n)$ for some $\lambda\wedge$-types $\tilde{\rho}_1, ..., \tilde{\rho}_n$. By the induction hypothesis there exists a $\tilde{\Delta}_1$ such that $\tilde{\Delta}_1 \vdash^{P}_{\lambda\wedge} M_1 : [\tilde{\rho}_1, ..., \tilde{\rho}_n] \to \tilde{\sigma}$ and $\mathrm{tr}^*(\tilde{\Delta}_1) \equiv \Gamma$. Therefore, using Lemma 4.2(ii), we have $\tilde{\Gamma} + \tilde{\Delta}_1 \vdash^{P}_{\lambda\wedge} M_1 M_2 : \tilde{\sigma}$.

Subcase 2.2. $\mathrm{left}(M_1 M_2) \equiv M_1\, \mathrm{left}(M_2)$. It is easily proved if we note that M_1 is in normal form and that M_1 is either a variable, an application, or a type application.

Subcase 2.3. $M \equiv (\lambda x : \rho.M_3)\, M_2$ and $\mathrm{left}(M) \equiv M_3[x := M_2]$. In this case, Lemma 4.4 is used. Assume that

x is not in Γ. The derivation of $\Gamma \vdash_{\lambda 2} M: \sigma$ ends with the following form:

$$\frac{\dfrac{\Gamma, x: \rho \vdash_{\lambda 2} M_3: \sigma}{\Gamma \vdash_{\lambda 2} (\lambda x: \rho . M_3): \rho \to \sigma} \qquad \Gamma \vdash_{\lambda 2} M_2: \rho}{\Gamma \vdash_{\lambda 2} (\lambda x: \rho . M_3)\, M_2: \sigma}$$

Since $\mathrm{bd}(M_2) < \mathrm{bd}(M)$, by assumption we have $\tilde{\Delta}_1 \vdash^P_{\lambda\wedge} M_2: \tilde{\rho}$ for some $\tilde{\Delta}_1$ and $\tilde{\rho}$ such that $\mathrm{tr}^*(\tilde{\Delta}_1) \equiv \Gamma$ and $\mathrm{tr}^*(\tilde{\rho}) \equiv \rho$. There are given $\tilde{\Gamma} \vdash^P_{\lambda\wedge} M_3[x := M_2]: \tilde{\sigma}$ and $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$. Let $\tilde{\Delta} \equiv \tilde{\Gamma} + \tilde{\Delta}_1$. Then, by Lemma 4.2(ii),

$$\tilde{\Delta} \vdash^P_{\lambda\wedge} M_2: \rho \qquad \text{and} \qquad \tilde{\Delta} \vdash^P_{\lambda\wedge} M_3[x := M_2]: \tilde{\sigma}.$$

As remarked at the beginning, $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma$, so that

$$\mathrm{tr}^*(\tilde{\Delta}), x: \mathrm{tr}^*(\tilde{\rho}) \vdash_{\lambda 2} M_3: \mathrm{tr}^*(\tilde{\sigma}).$$

By condition (4), $\mathrm{FTV}(\rho) \subseteq \mathrm{FTV}(M) \subseteq P$. Now all the conditions of Lemma 4.4 are satisfied, and we obtain $\lambda\wedge$-types $\tilde{\rho}_1, \ldots, \tilde{\rho}_n$ such that

$$\tilde{\Delta}, x: [\tilde{\rho}_1, \ldots, \tilde{\rho}_n] \vdash^P_{\lambda\wedge} M_3: \tilde{\sigma}$$

and

$$\tilde{\Delta} \vdash^P_{\lambda\wedge} M_2: \tilde{\rho}_i \qquad (1 \leqslant i \leqslant n).$$

Hence, $\tilde{\Delta} \vdash^P_{\lambda\wedge} (\lambda x: \rho . M_3) M_2: \tilde{\sigma}$.

Case 3. $M \equiv \Lambda t . M_1$. Assume that t is not free in Γ. Then, since $\Gamma \vdash_{\lambda 2} M: \sigma$, we have $\Gamma \vdash_{\lambda 2} M_1: \rho$ for some ρ such that $\forall t . \rho \equiv \sigma$. The derivation of $\tilde{\Gamma} \vdash_{\lambda\wedge} \mathrm{left}(M): \tilde{\sigma}$ ends with the form:

$$\frac{\tilde{\Gamma} \vdash_{\lambda\wedge} M_2: \tilde{\sigma}_1 \qquad \tilde{\Gamma} \vdash_{\lambda\wedge} M_2[s := \alpha]: \tilde{\sigma}_2}{\tilde{\Gamma} \vdash_{\lambda\wedge} \Lambda s . M_2: \langle s, \tilde{\sigma}_1, \tilde{\sigma}_2 \rangle}$$

where $\Lambda s . M_2 \equiv \mathrm{left}(M)$ and $\langle s, \tilde{\sigma}_1, \tilde{\sigma}_2 \rangle \equiv \tilde{\sigma}$. Let $P' = P \cup \{s\}$. By definition, $\tilde{\Gamma} \vdash^{P'}_{\lambda\wedge} M_2: \tilde{\sigma}_1$. We may assume that $t \equiv s$. Thus, $\mathrm{FTV}(M_1) \subseteq \mathrm{FTV}(M) \cup \{s\} \subseteq P'$, since $\mathrm{FTV}(M) \subseteq P$. Therefore, by the induction hypothesis, there exist $\tilde{\Delta}_1$ and $\tilde{\tau}_1$ such that $\tilde{\Delta}_1 \vdash^{P'}_{\lambda\wedge} M_1: \tilde{\tau}_1$ and $\mathrm{tr}^*(\tilde{\Delta}_1) \equiv \Gamma$. Let u be a type variable not occurring in the derivation of $\tilde{\Delta}_1 \vdash^{P'}_{\lambda\wedge} M_1: \tilde{\tau}_1$. Then obviously

$$\tilde{\Delta}_1[s := u] \vdash^{P'}_{\lambda\wedge} M_1[s := u]: \tilde{\tau}_1[s := u]$$

and $\mathrm{tr}^*(\Delta_1[s := u]) \equiv \mathrm{tr}^*(\Delta_1) \equiv \Gamma$ since s is not free in Γ. If we take $\tilde{\Delta} \equiv \tilde{\Delta}_1 + \tilde{\Delta}_1[s := u]$, then by Lemma 4.1(ii) we have $\tilde{\Delta} \vdash^{P'}_{\lambda\wedge} M_2: \tilde{\tau}_1$ and $\tilde{\Delta} \vdash^{P'}_{\lambda\wedge} M_1[s := u]: \tilde{\tau}[s := u]$, and therefore $\tilde{\Delta} \vdash^P_{\lambda\wedge} \Lambda s . M_1: \langle s, \tilde{\tau}_1, \tilde{\tau}_1[s := u] \rangle$.

Case 4. $M \equiv M_1 \alpha$. We distinguish two subcases.

Subcase 4.1. $\mathrm{left}(M_1 \alpha) \equiv \mathrm{left}(M_1)\, \alpha$. Since $\Gamma \vdash_{\lambda 2} M: \sigma$, we have $\Gamma \vdash_{\lambda 2} M_1: \forall t . \rho$ for some $\forall t . \rho$ such that $\rho[t := \alpha] \equiv \sigma$. Since $\tilde{\Gamma} \vdash^P_{\lambda\wedge} \mathrm{left}(M_1)\, \alpha: \tilde{\sigma}$, we have $\tilde{\Gamma} \vdash^P_{\lambda\wedge} \mathrm{left}(M_1): \langle s, \tilde{\rho}, \tilde{\sigma} \rangle$ for some s and $\tilde{\rho}$ such that $\mathrm{tr}^*(\tilde{\rho})[s := \alpha] \equiv \mathrm{tr}^*(\tilde{\sigma})$. By definition, M_1 is in the form of an application or a type application, and $\mathrm{FTV}(M_1) \subseteq \mathrm{FTV}(M) \subseteq P$. Therefore, by the induction hypothesis, we have $\tilde{\Delta} \vdash^P_{\lambda\wedge} M_1: \langle s, \tilde{\rho}, \tilde{\sigma} \rangle$ for some $\tilde{\Delta}$ such that $\mathrm{tr}^*(\tilde{\Delta}) \equiv \Gamma$. Hence, $\tilde{\Delta} \vdash^P_{\lambda\wedge} M_1 \alpha: \tilde{\sigma}$.

Subcase 4.2. $M \equiv (\Lambda t . M_2)\, \alpha$ and $\mathrm{left}(M) \equiv M_2[t := \alpha]$. The derivation of $\Gamma \vdash_{\lambda 2} M: \sigma$ ends with the following form:

$$\frac{\dfrac{\Gamma \vdash_{\lambda 2} M_2: \rho}{\Gamma \vdash_{\lambda 2} \Lambda t . M_2: \forall t . \rho}}{\Gamma \vdash_{\lambda 2} (\Lambda t . M_2)\, \alpha: \rho[t := \alpha]}$$

where $\rho[t := \alpha] \equiv \sigma$ and t is not free in Γ. Let $P' = P \cup \{t\}$. Since $\mathrm{bd}(M_2) < \mathrm{bd}(M)$, by assumption we have $\tilde{\Delta}_1 \vdash^{P'}_{\lambda\wedge} M_2: \tilde{\rho}$ for some $\tilde{\Delta}_1$ and $\tilde{\rho}$ such that $\mathrm{tr}^*(\tilde{\Delta}_1) \equiv \Gamma$ and $\mathrm{tr}^*(\tilde{\rho}) \equiv \rho$. There are given $\tilde{\Gamma} \vdash^P_{\lambda\wedge} M_2[t := \alpha]: \tilde{\sigma}$ and $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$. Let $\tilde{\Delta} \equiv \tilde{\Gamma} + \tilde{\Delta}_1$. Then, by Lemma 4.2(ii), $\tilde{\Delta} \vdash^{P'}_{\lambda\wedge} M_2: \tilde{\rho}$ and $\tilde{\Delta} \vdash^P_{\lambda\wedge} M_2[t := \alpha]: \tilde{\sigma}$. As remarked at the beginning, $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma \equiv \mathrm{tr}^*(\tilde{\rho})[t := \alpha]$, and we may assume $t \notin P \cup \mathrm{FTV}(\mathrm{tr}^*(\sigma))$. Therefore we can construct the following derivation tree:

$$\frac{\dfrac{\tilde{\Delta} \vdash_{\lambda\wedge} M_2: \tilde{\rho} \qquad \tilde{\Delta} \vdash_{\lambda\wedge} M_2[t := \alpha]: \tilde{\sigma}}{\tilde{\Delta} \vdash_{\lambda\wedge} \Lambda t . M_2: \langle t, \tilde{\rho}, \tilde{\sigma} \rangle}}{\tilde{\Delta} \vdash_{\lambda\wedge} (\Lambda t . M_2) \alpha: \tilde{\sigma}} \quad \blacksquare$$

Finally, combining Lemmas 4.3 and 4.5, we conclude with Theorem 4.1 as follows.

Proof of Theorem 4.1. We use induction on $\mathrm{bd}(M)$. Suppose M is not in normal form; otherwise, it is just Lemma 4.3. We may assume $\mathrm{FTV}(M) \subseteq P$. Since $\mathrm{bd}(\mathrm{left}(M)) < \mathrm{bd}(M)$, by the induction hypothesis we have $\tilde{\Gamma} \vdash^P_{\lambda\wedge} \mathrm{left}(M): \tilde{\sigma}$ for some $\tilde{\Gamma}$ and $\tilde{\sigma}$ such that $\mathrm{tr}^*(\tilde{\Gamma}) \equiv \Gamma$ and $\mathrm{tr}^*(\tilde{\sigma}) \equiv \sigma$. Therefore, by Lemma 4.5, there exist $\tilde{\Delta}$ and $\tilde{\tau}$ such that $\tilde{\Delta} \vdash^P_{\lambda\wedge} M: \tilde{\tau}$, $\mathrm{tr}^*(\tilde{\Delta}) \equiv \Gamma$, and $\mathrm{tr}^*(\tilde{\tau}) \equiv \sigma$.

The second half of the theorem is clear from that of Lemma 4.5. Note that if the normal form of M is either a variable, an application, or a type application, then so is M. $\blacksquare$

5. RESTRICTED SYSTEM

We have shown that typings in $\mathbf{T}_{\wedge *}$ can be translated into $\mathbf{T}_\forall$ by tr(—), and conversely that typings in $\mathbf{T}_\forall$ can be embedded into $\mathbf{T}_{\wedge *}$. These results suggest that some properties of $\mathbf{T}_\wedge$ might be translated into $\mathbf{T}_\forall$ if we fill the gap between $\mathbf{T}_\wedge$ and $\mathbf{T}_{\wedge *}$. We should recall that the intersection introduction rule $(\wedge I)^*$ of $\mathbf{T}_{\wedge *}$ is restricted. In this section we show that the condition on $(\wedge I)^*$ can be removed when we consider the rank 2 fragments $\mathbf{T}_{\forall r2}$ of $\mathbf{T}_\forall$ and $\mathbf{T}_{\wedge r2}$ of $\mathbf{T}_\wedge$, which have been introduced by Leivant (1983). Namely, our main theorems still hold for $\mathbf{T}_{\forall r2}$ and $\mathbf{T}_{\wedge r2}$ without any restriction of $(\wedge I)$. This means that typing in $\mathbf{T}_{\forall r2}$ can be completely handled in $\mathbf{T}_{\wedge r2}$ by tr(—).

As an application, we can show that for each λ-type M all $\forall$-types of M in $\mathbf{T}_{\forall r2}$ can be characterized simply using intersection types. All $\wedge$-types of M in $\mathbf{T}_{\wedge r2}$ are characterized in a way similar to principal typing for the ML type system (Hindley, 1969; Milner, 1978; Damas and Milner, 1982), which is easily shown by the same method as described in Ronchi Della Rocca and Venneri (1984) and Ronchi Della Rocca (1988). More precisely, for each λ-term M, if M is typable in $\mathbf{T}_{\wedge r2}$, then there exists a $\wedge$-type σ, called the principal $\wedge$-type, such that all $\wedge$-types of M are obtained from σ by applying substitution for type variables and an additional operation such as *expansion*, introduced by Coppo *et al.* (1980) and Ronchi Della Rocca and Venneri (1984). Translating this result into $\mathbf{T}_{\forall r2}$, we have a similar characterization for $\mathbf{T}_{\forall r2}$. Namely, for each λ-term M, if M is typable in $\mathbf{T}_{\forall r2}$, then M has the principal $\wedge$-type σ for $\mathbf{T}_{\wedge r2}$ and all $\forall$-types of M in $\mathbf{T}_{\forall r2}$ are obtained from σ by applying substitution and expansion and then by applying tr(—). Furthermore there exists a procedure taking a λ-term M and producing the principal $\wedge$-type of M if one exists. The algorithm has been implicitly presented by Leivant(1983). However, the decidability itself has been proved already. Kfoury and Tiuryn (1992), in a different way, have shown that it is decidable whether a given λ-term is typable in $\mathbf{T}_{\forall r2}$. They used a translation between $\mathbf{T}_{\forall r2}$ and the ML type system. See also Tiuryn (1990) for the type inference problems.

In the rest of this section, we provide the formal definition of $\mathbf{T}_{\wedge r2}$ and show that the condition on $(\wedge I)^*$ can be removed in case of $\mathbf{T}_{\wedge r2}$. Our definition of the rank 2 system $\mathbf{T}_{\wedge r2}$ is slightly different from the definition proposed by Leivant (1983), but they are essentially the same.

The system $\mathbf{T}_{\wedge r2}$ is defined by restricting $\wedge$-types. A *simple* type is a type generated from type variables by applying the type constructor $\rightarrow$. A *generic* $\wedge$-type is an $\wedge$-type obtained from simple types by applying the type constructor $\wedge$ finitely many times. Typing statements in $\mathbf{T}_{\wedge r2}$ are restricted to the form

$$x_1 : \sigma_1, ..., x_n : \sigma_n \vdash_{\wedge r2} M : \rho_1 \rightarrow \cdots \rightarrow \rho_m \rightarrow \tau,$$

where $\sigma_1, ..., \sigma_n, \rho_1, ..., \rho_m$, and τ are generic $\wedge$-types. The inference rules are defined as follows.

DEFINITION (Inference Rules of $\mathbf{T}_{\wedge r2}$). Let Γ be a basis $\langle x_1 : \sigma_1, ..., x_n : \sigma_n \rangle$ such that $\sigma_1, ..., \sigma_n$ are generic $\wedge$-types. Let $\sigma, \tau, \rho_1, ..., \rho_m$ be generic $\wedge$-types. We have the rules

$$(var) \qquad \Gamma \vdash_{\wedge r2} x_i : \sigma_i \quad (1 \leqslant i \leqslant n)$$

$$(\rightarrow I) \qquad \frac{\Gamma, x : \sigma \vdash_{\wedge r2} M : \rho_1 \rightarrow \cdots \rightarrow \rho_m \rightarrow \tau}{\Gamma \vdash_{\wedge r2} \lambda x . M : \sigma \rightarrow \rho_1 \rightarrow \cdots \rightarrow \rho_m \rightarrow \tau}$$

$$(\rightarrow E) \qquad \frac{\Gamma \vdash_{\wedge r2} M : \sigma \rightarrow \rho_1 \rightarrow \cdots \rightarrow \rho_m \rightarrow \tau \quad \Gamma \vdash_{\wedge r2} N : \sigma}{\Gamma \vdash_{\wedge r2} MN : \rho_1 \rightarrow \cdots \rightarrow \rho_m \rightarrow \tau}$$

$$(\wedge I) \qquad \frac{\Gamma \vdash_{\wedge r2} M : \sigma \qquad \Gamma \vdash_{\wedge r2} M : \tau}{\Gamma \vdash_{\wedge r2} M : \sigma \wedge \tau}$$

$$(\wedge E) \qquad \frac{\Gamma \vdash_{\wedge r2} M : \sigma \wedge \tau}{\Gamma \vdash_{\wedge r2} M : \sigma} \qquad \frac{\Gamma \vdash_{\wedge r2} M : \sigma \wedge \tau}{\vdash_{\wedge r2} M : \tau}$$

LEMMA 5.1. *If $\Gamma \vdash_{\wedge r2} M : \sigma$ and $\Gamma \vdash_{\wedge r2} M : \tau$, then there exist Δ and ρ such that $\Delta \vdash_{\wedge r2} M : \rho$,* $\mathrm{tr}(\Gamma) \equiv \mathrm{tr}(\Delta)$, $\forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\sigma)$, *and* $\forall t_1 \cdots \forall t_n . \mathrm{tr}(\rho) \sqsubseteq \mathrm{tr}(\tau)$ *for some type variables $t_1, ..., t_n$ not free in* $\mathrm{tr}(\Delta)$.

Proof. First we introduce an operation on $\wedge$-types similar to cq(—, —) used in the proof of Lemma 2.1. Let θ be a injective mapping from the set of all $\wedge$-types into the set of all type variables. In general, for each pair of $\wedge$-types σ and τ we define the $\wedge$-type $\mathrm{ci}_\theta(\sigma, \tau)$ as follows:

$$\mathrm{ci}_\theta(\sigma, \tau) \equiv \begin{cases} t \\ \quad \text{if} \quad \sigma \equiv \tau \equiv t \quad \text{(type variable)}, \\ \mathrm{ci}_\theta(\sigma_1, \tau_1) \rightarrow \mathrm{ci}_\theta(\sigma_2, \tau_2) \\ \quad \text{if} \quad \sigma \equiv \sigma_1 \rightarrow \sigma_2 \quad \text{and} \quad \tau \equiv \tau_1 \rightarrow \tau_2, \\ \mathrm{ci}_\theta(\sigma_1, \tau) \wedge \mathrm{ci}_\theta(\sigma_2, \tau) \\ \quad \text{if} \quad \sigma \equiv \sigma_1 \wedge \sigma_2, \\ \mathrm{ci}_\theta(\sigma, \tau_1) \wedge \mathrm{ci}_\theta(\sigma, \tau_2) \\ \quad \text{if} \quad \sigma \not\equiv \sigma_1 \wedge \sigma_2 \quad \text{and} \quad \tau \equiv \tau_1 \wedge \tau_2, \\ \theta(\sigma, \tau) \\ \quad \text{otherwise.} \end{cases}$$

We can easily prove the following three statements.

(i) If $x_1 : \sigma_1, ..., x_n : \sigma_n \vdash_{\wedge r2} M : \sigma$ and $x_1 : \tau_1, ..., x_n : \tau_n \vdash_{\wedge r2} M : \tau$, then

$$x_1 : \mathrm{ci}_\theta(\sigma_1, \tau_1), ..., x_n : \mathrm{ci}_\theta(\sigma_n, \tau_n) \vdash_{\wedge r2} M : \mathrm{ci}_\theta(\sigma, \tau).$$

(ii) For any pair of generic $\wedge$-types σ and τ, if the range of θ contains no type variables occuring in σ or τ, then $\mathrm{tr}(\sigma \wedge \tau) \cong \forall t_1 \cdots \forall t_n . \mathrm{tr}(\mathrm{ci}_\theta(\sigma, \tau))$, where $t_1, ..., t_n$ are newly introduced in $\mathrm{ci}_\theta(\sigma, \tau)$ by applying ci_θ(—, —).

(iii) $\mathrm{tr}(\sigma) \equiv \mathrm{tr}(\mathrm{ci}_\theta(\sigma, \sigma))$ for any generic $\wedge$-type σ.

We now prove the present lemma. Suppose that $\Gamma \vdash_{\wedge r2} M : \sigma$ and $\Gamma \vdash_{\wedge r2} M : \tau$. We may assume that the range of θ contains no type variables occurring in the derivations of those two typings. Define $\Delta \equiv \langle x_1 : \mathrm{ci}_\theta(\rho_1, \rho_1), ..., x_m : \mathrm{ci}_\theta(\rho_m, \rho_m) \rangle$, where $\Gamma \equiv \langle x_1 : \rho_1, ..., x_m : \rho_m \rangle$. Let $\rho \equiv \mathrm{ci}_\theta(\sigma, \tau)$, and let $t_1, ..., t_n$ be all type variables newly introduced in ρ by the application of ci_θ(—, —). Then, using (i), (ii), and (iii) described above, we can easily prove that Δ, ρ, and $t_1, ..., t_n$ satisfy the conditions stated in the lemma. ∎

We next define $\mathbf{T}_{\forall r2}$ and show that $\mathbf{T}_{\forall r2}$ and $\mathbf{T}_{\wedge r2}$ are essentially equivalent. The system $\mathbf{T}_{\forall r2}$ is defined by restricting $\forall$-types in a similar way to $\mathbf{T}_{\wedge r2}$. A *generic* $\forall$-type is a $\forall$-type of the form $\forall t_1 \cdots \forall t_n.\sigma$, where σ is a simple type. Typings in $\mathbf{T}_{\forall r2}$ are restricted to the form

$$x_1: \sigma_1, ..., x_n: \sigma_n \vdash_{\forall r2} M: \rho_1 \to \cdots \to \rho_m \to \tau,$$

where $\sigma_1, ..., \sigma_n, \rho_1, ..., \rho_m, \tau$ are generic $\forall$-types. The inference rules are defined as follows:

DEFINITION (Inference Rules of $\mathbf{T}_{\forall r2}$). Let Γ be a basis $\langle x_1: \sigma_1, ..., x_n: \sigma_n \rangle$ such that $\sigma_1, ..., \sigma_n$ are generic $\forall$-types. Let $\sigma, \tau, \rho_1, ..., \rho_m$ be generic $\forall$-types. We have the rules

$$(var) \qquad \Gamma \vdash_{\forall r2} x_i: \sigma_i \qquad (1 \leqslant i \leqslant n)$$

$$(\to I) \qquad \frac{\Gamma, x: \sigma \vdash_{\forall r2} M: \rho_1 \to \cdots \to \rho_m \to \tau}{\Gamma \vdash_{\forall r2} \lambda x.M: \sigma \to \rho_1 \to \cdots \to \rho_m \to \tau}$$

$$(\to E) \qquad \frac{\Gamma \vdash_{\forall r2} M: \sigma \to \rho_1 \to \cdots \to \rho_m \to \tau \quad \Gamma \vdash_{\forall r2} N: \sigma}{\vdash_{\forall r2} MN: \rho_1 \to \cdots \to \rho_m \to \tau}$$

$$(\forall I) \qquad \frac{\Gamma \vdash_{\forall r2} M: \sigma}{\Gamma \vdash_{\forall r2} M: \forall t.\sigma}$$

where t is not free in Γ;

$$(\wedge E) \qquad \frac{\Gamma \vdash_{\forall r2} M: \forall t.\sigma}{\Gamma \vdash_{\forall r2} M: \sigma[t := \alpha]}$$

where α is a simple type.

LEMMA 5.2. (i) *If* $\Gamma \vdash_{\wedge r2} M: \sigma$, *then* $\mathrm{tr}(\Gamma) \vdash_{\forall r2} M: \mathrm{tr}(\sigma)$.

(ii) *If* $\Delta \vdash_{\forall r2} M: \tau$, *then* $\Gamma \vdash_{\wedge r2} M: \sigma$ *for some* Γ *and* σ *such that* $\mathrm{tr}(\Gamma) \equiv \mathrm{reg}(\Delta)$ *and* $\mathrm{tr}(\sigma) \equiv \mathrm{reg}(\tau)$.

Proof. (i) Clear from Lemma 5.1 and Theorem 2.5.

(ii) Straightforward by induction on the structure of M. ∎

ACKNOWLEDGMENTS

This paper was written while the author was visiting LFCS, Department of Computer Science, University of Edinburgh. He would like to thank Professor Rod Burstall for his encouragement during the visit.

Received January 16, 1992; final manuscript received June 28, 1993

REFERENCES

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

Barendregt, H. (1992), Lambda calculi with types, *in* "Handbook of Logic in Computer Science, Vol. II" (S. Abramsky, D. M. Gabbai, and T. S. E. Maibaum, Eds.), Oxford Univ. Press, London.

Cardone, F., and Coppo, M. (1990), Two extensions of Curry's type inference system, *in* "Logic and Computer Science" (P. Odifreddi, Ed.), pp. 19–75, Academic Press, London.

Coppo, M., and Dezani-Ciancaglini, M. (1980), An extension of basic functionality theory for λ-calculus, *Notre Dame J. Formal Logic* **21**, 685–693.

Coppo, M., Dezani-Ciancaglini, M., and Venneri, B. (1980), Principal type schemes and λ-calculus semantics, *in* "To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism" (J. P. Seldin and J. R. Hindley, Eds.), pp. 535–560, Academic Press, London.

Coppo, M., Dezani-Ciancaglini, M., and Venneri, B. (1981), Functional characters of solvable terms, *Z. Math. Logik. Grundlag. Math.* **27**, 45–58.

Curry, H. B., and Feys, R. (1958), "Combinatory Logic, Vol. I," North-Holland, Amsterdam.

Damas, L., and Milner, R. (1982), Principal type-schemes for functional programs, *in* "Proceedings, 9th ACM Symposium on Principles of Program Languages," pp. 207–212.

Fortune, S., Leivant, D., and O'Donnell, M. (1983), The expressiveness of simple and second-order type structures, *J. Assoc. Comput. Mach.* **30** (1), 151–185.

Giannini, P., and Ronchi Della Rocca, S. (1988), Characterization of typings in polymorphic type discipline, *in* "Proceedings, 3rd IEEE Symposium on Logic in Computer Science," pp. 61–70.

Girard, J.-Y. (1972), "Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur," Doctoral Thesis, University of Paris VII.

Girard, J.-Y., Lafont, Y., and Taylor, P. (1989), "Proofs and Types," Cambridge University Press, Cambridge.

Gödel, K. (1958), Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes, *Dialectica* **12**, 76–82.

Hindley, R. (1969), The principal type-scheme of an object in combinatory logic, *Trans. Am. Math. Soc.* **146**, 29–60.

Huet, G. (1976), "Résolution d'équations dans des languages d'ordre 1, 2, ..., ω," Doctoral Thesis, University of Paris VII.

Kfoury, A. J., and Tiuryn, J. (1992), Type reconstruction in finite rank fragments of the second-order λ-calculus, *Inform. and Comput.* **98**, 228–257.

Lassez, J.-L., Maher, M. J., and Marriott, K. G. (1986), Unification Revisited, Research Report, RC 12394 (#55630), IBM T. J. Watson Research Center.

Leivant, D. (1983), Polymorphic type inference, *in* "Proceedings, 10th ACM Symposium on Principles of Programming Languages," pp. 88–98.

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

Leivant, D. (1990), Discrete polymorphism, *in* "Proceedings, 17th ACM Symposium on Principles of Programming Languages," pp. 288–297.

Milner, R. (1978), A theory of type polymorphism in programming, *J. Comput. System Sci.* **17**, 348–375.

Mitchell, J. C. (1988), Polymorphic type inference and containment, *Inform. and Comput.* **76**, 211–249.

Plotkin, G. D. (1970), A note on inductive generalization, *in* "Machine Intelligence 5" (B. Meltzer and D. Michie, Eds.), pp. 153–163, Edinburgh Univ. Press, Edinburgh.

Pottinger, G. (1980), A type assignment for the strongly normalizable λ-terms, *in* "To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism" (J. P. Seldin and J. R. Hindley, Eds.), pp. 561–577, Academic Press, London.

Prawitz, D. (1965), "Natural Deduction, a Proof-Theoretic Study," Almqvist & Wiksell, Stockholm.

Reynolds, J. C. (1970), Transformational system and the algebraic structure of atomic formula, *in* "Machine Intelligence 5" (B. Meltzer and D. Michie, Eds.), pp. 135–151, Edinburgh Univ. Press, Edinburgh.

Reynolds, J. C. (1974), Towards a theory of type structure, *in* "Proceedings, Programming Symposium" (B. Robinet, Ed.), pp. 408–425, Lecture Notes in Computer Science, Vol. 19, Springer-Verlag, Berlin.

Ronchi Della Rocca, S., and Venneri, B. (1984), Principal type schemes for an extended type theory, *Theoret. Comput. Sci.* **28**, 151–169.

Ronchi Della Rocca, S. (1988), Principal type scheme and unification for intersection type discipline, *Theoret. Comput. Sci.* **59**, 181–209.

Tiuryn, J. (1990), Type inference problems: A survey, *in* "Proceedings, Mathematical Foundations of Computer Science" (B. Rovan, Ed.), pp. 105–120, Lecture Notes in Computer Science, Vol. 452, Springer-Verlag, Berlin.