
UNIQUENESS TYPING FOR INTERSECTION TYPES

RICHARD STATMAN, ANDREW POLONSKY

ABSTRACT. Working in a variant of the intersection type assignment system of Coppo, Dezani-Ciancaglini and Veneri (CDV) [5], we prove several facts about sets of terms having a given intersection type. Our main result is that every strongly normalizing term M admits a *uniqueness typing*, which is a pair (Γ, A) such that

- $\Gamma \vdash M : A$
- $\Gamma \vdash N : A \implies M =_{\beta\eta} N$

We also discuss several presentations of intersection type algebras, and the corresponding choices of type assignment rules.

1. INTRODUCTION

Since their introduction, intersection types have played a role of increasing prominence in programming languages research. From completeness of type assignment [3], to characterization of strongly normalizing (and weakly normalizing) terms [6], to syntactic presentations of domain models [1], including graph models and filter models [8], to classifications of certain classes of easy terms [4], to using types to count resources [7], and many others — the theory of intersection types is now a well-established research field of type theory.

Although enormously more expressive than simple types, intersection types enjoy most of the fundamental properties expected of type systems, including stability under substitution and reduction (Subject Reduction Theorem), decidability of type-checking, and certain forms of principal typing.

In the present paper, we prove that, among many intersection types a term may have, there will always be one for which the term is the only inhabitant, up to beta-eta equality.

The paper is organized as follows. First, we review the syntactic theory of intersection types. Next, we discuss the concept of intersection type algebras from several perspectives, with the goal of obtaining a unique representation of every intersection type. This leads us to the alternative formulation of intersection type assignment based on “essential intersection types” introduced by van Bakel [10]. Here we also establish a technical lemma to be used in the proof of the main result. Finally, we prove the uniqueness typing theorem in a sequence of progressively more general forms.

Key words and phrases: Intersection types, Uniqueness typing, Lambda Calculus.

2. INTERSECTION TYPE ASSIGNMENT SYSTEM

We work in the system CDV of intersection types without the top element ω . CDV was originally introduced by [5] to obtain a type-theoretic characterization of solvable terms. This system has several variants in the literature.

The modern presentation [2] treats intersection \cap as a binary type constructor on par with the arrow type \rightarrow . The set of types generated by these is then imbued with a pre-order relation that is used in the subsumption rule. This formulation is convenient for the construction of filter models.

In the original paper [5], intersections and types belonged to different grammar sorts, which were defined by mutual recursion. This formulation is convenient when types are used for syntactic analysis of terms.

Yet another presentation, due to van Bakel [10], was introduced in his paper *essential intersection types*. Here the type assignment rules are restricted as far as possible to remove all redundancies. This system is most convenient for a proof-theoretic analysis of typability, and is the one we will make use of in the proof of our main result.

The equivalence of these systems is shown in [10]. Rather than reproduce the proof here, we will give a brief review of the systems, which should make the relationship between them clear to the reader. A particular aspect to note is how the choice of type assignment rules relates to the presentation of the underlying intersection type theory.

2.1. Intersection as a type constructor. We begin with the formulation in [2].

Let \mathbb{A} be a countable set whose elements are called *type atoms*. The set of intersection types over \mathbb{A} is given by the following grammar

$$A \in \mathbb{T} \quad ::= \quad \mathbb{A} \mid \mathbb{T} \rightarrow \mathbb{T} \mid \mathbb{T} \cap \mathbb{T}$$

Definition 2.1. The types are considered together with a preorder generated by the following axioms and rules. This preorder relation will be used in the rules of type assignment.

$$\begin{array}{c} \frac{}{A \leq A} \quad \frac{}{A \cap B \leq A} \quad \frac{}{A \cap B \leq B} \quad \frac{}{(A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow (B \cap C)} \\ \frac{A \leq B \quad B \leq C}{A \leq C} \quad \frac{C \leq A \quad C \leq B}{C \leq A \cap B} \quad \frac{A' \leq A \quad B \leq B'}{A \rightarrow B \leq A' \rightarrow B'} \end{array}$$

Let \mathbb{V} be a countable set whose elements are called *term variables*. The set of lambda terms is generated by the grammar

$$M \in \mathbb{\Lambda} \quad ::= \quad \mathbb{V} \mid \mathbb{\Lambda} \mathbb{\Lambda} \mid \lambda \mathbb{V}. \mathbb{\Lambda}$$

A *context* is a finite function $\Gamma : \mathbb{V} \rightarrow \mathbb{T}$. Contexts are denoted as $\Gamma = \{x_1 : A_1, \dots, x_k : A_k\}$. We write $\mathcal{C}\mathcal{X}\mathcal{T}$ for the set of all contexts.

We define the ternary *typing relation* $(- \vdash - : -) \subseteq \mathcal{C}\mathcal{X}\mathcal{T} \times \mathbb{\Lambda} \times \mathbb{T}$ by the following set of inference rules. These include the rules of the simply typed lambda calculus:

$$\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \quad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x. M : A \rightarrow B}$$

together with two more rules treating intersection and subsumption:

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \cap B} \quad \frac{\Gamma \vdash M : A \quad A \leq B}{\Gamma \vdash M : B}$$

Note that the typability relation on terms is dependent on the type preorder \leq . Below, we will analyze several ways of generating this preorder.

2.2. Intersection type algebras. The following concept is called an “extended abstract type structure” in [1].

Definition 2.2. An *intersection type algebra* (ita) is a structure $\mathcal{A} = (T, \leq, \cap, \Rightarrow)$, where (T, \leq, \cap) is a meet semilattice and $\Rightarrow : (T, \geq) \times (T, \leq) \rightarrow (T, \leq)$ is a binary operation on T that is antimonotone in the first argument and monotone in the second, and furthermore, for each $x \in T$, the map $(x \Rightarrow -) : (T, \leq, \cap) \rightarrow (T, \leq, \cap)$ preserves meets.

Definition 2.3. An *intersection type preorder* (itp) is a structure $\mathcal{A} = (P, \leq, \cap, \Rightarrow)$, where (P, \leq) is a preorder, $x \cap y$ is a maximal lower bound of x and y , \Rightarrow is as above, and $a \Rightarrow x \cap y$ is a maximal lower bound of $a \Rightarrow x$ and $a \Rightarrow y$ for all x and y .

If $(P, \leq, \cap, \Rightarrow)$ is an intersection type preorder, then the equivalence relation

$$x \sim y \quad := \quad x \leq y \quad \text{and} \quad y \leq x$$

is a congruence with respect to \cap and \Rightarrow . The quotient P/\sim then has the structure of an intersection type algebra.

At the same time, all of the standard examples, including those below, will indeed be partial orders, with \leq antisymmetric. For our purposes, it will therefore suffice to restrict attention to intersection type algebras.

Examples 2.4. (1) Let D be a λ -model, combinatory algebra, or a general applicative structure (magma). The powerset $\wp(D)$ carries the structure of an ita, where

$$\begin{aligned} X \leq Y &\iff X \subseteq Y \\ X \wedge Y &= X \cap Y \\ X \Rightarrow Y &= \{d \in D \mid \forall x \in X. dx \in Y\} \end{aligned}$$

- (2) Every Heyting Algebra is an ita, since Heyting implication is antimonotone in its first argument, and monotone and \wedge -preserving in the second.
- (3) Every lattice-ordered group (AKA ℓ -group) $\mathcal{G} = (G, \leq, \wedge, \vee, \cdot, e, (-)^{-1})$ is an ita, by taking the semilattice to be inherited from the order, and defining

$$a \rightarrow b \quad := \quad a^{-1}b$$

The distributive law follows since

$$a \rightarrow (b \wedge c) = a^{-1}(b \wedge c) = a^{-1}b \wedge a^{-1}c = (a \rightarrow b) \wedge (a \rightarrow c)$$

Similarly, $a \leq a'$ implies

$$a \rightarrow b = a^{-1}b \geq (a')^{-1}b = a' \rightarrow b$$

- (4) The tropical semiring $\mathcal{Z} = (\mathbb{Z}, \min, +)$ is an ita, as is the ℓ -group $\mathcal{Z}[\vec{x}]$ of semiring polynomials with variables in \vec{x} and coefficients in \mathcal{Z} .

The set of types \mathbb{T} is the *free* ita on the set \mathbb{A} . Thus, every “type environment” $\rho : \mathbb{A} \rightarrow \mathcal{A}$, where \mathcal{A} is an ita, extends uniquely to an ita homomorphism from \mathbb{T} into \mathcal{A} .

Moreover, this holds for any set of atoms \mathbb{A} . For example, if $\mathbb{A} = \{o\}$, then every type $A \in \mathbb{T}[o]$ gives rise to a $(\min, +)$ -polynomial $p_A(x) \in \mathcal{Z}[x]$ by sending o to x .

3. SOME PRESENTATIONS OF FREE INTERSECTION TYPE ALGEBRAS

We will now review several ways that the free ita on a set of generators can be obtained. This will enable us to eventually obtain a much more manageable set of representatives for the equivalence class of a type modulo the relation

$$A \sim B \iff A \leq B \ \& \ B \leq A \quad (3.1)$$

3.1. Inequational. The most obvious way to get the free ita on a given set \mathbb{A} is to do precisely what was already done in subsection 2.1: The carrier of the ita is \mathbb{T}/\sim , where \leq is given by the rules of Definition 2.1, and \sim is (3.1).

This tautologically results in a free ita on the set \mathbb{A} .

3.2. Equational. Alternatively, we can make use of the fact that the concept of ita is completely algebraic. Using the equivalence

$$x \leq y \iff x \wedge y = x \quad (3.2)$$

the meet semilattice part of the definition can be captured by the rules of an idempotent commutative semigroup (ICS):

$$x \wedge x = x \quad (\text{I})$$

$$x \wedge y = y \wedge x \quad (\text{C})$$

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z \quad (\text{S})$$

The laws concerning \Rightarrow can also be expressed equationally:

$$(x \rightarrow y) \wedge (x \rightarrow z) = x \rightarrow (y \wedge z) \quad (3.3)$$

$$(x \rightarrow y) \wedge (x \wedge z \rightarrow y) = (x \rightarrow y) \quad (3.4)$$

where the second law expresses the anti-monotonicity of \leq in the first argument.

Thus, the free ita can be seen as the set of all terms built from \mathbb{A} using the binary operations \rightarrow and \cap quotiented by the congruence generated by the equations above.

3.3. Rewriting-theoretic. Next, we could orient the above equations in an effort to obtain a convergent presentation. While the rules like commutativity prevent this goal from being fully realized, rewriting theory can offer useful insights into the structure of free itas (including intersection types), see [9].

Of particular interest is the operation of taking the normal form of a type $A \in \mathbb{T}$ with respect to the distributivity rule:

$$A \rightarrow (B \cap C) \longrightarrow (A \rightarrow B) \cap (A \rightarrow C) \quad (\text{dist})$$

Taking the *dist*-normal form (DNF) of $A \in \mathbb{T}$ results in a type expression that can be generated according to the two-phase grammar

$$A \in \mathbb{T}_{\rightarrow} ::= X_1 \rightarrow \cdots \rightarrow X_m \rightarrow \alpha \quad (3.5)$$

$$X \in \mathbb{T}_{\cap} ::= A_1 \cap \cdots \cap A_n \quad (3.6)$$

3.4. Proof-theoretic. Assuming only the ICS axioms as given, we can characterize inequality between intersection types that would result from adding the two remaining rules (the rightmost rules in Definition 2.1) by the following conditions.

Recall that an occurrence inside a type expression is called *positive* if it occurs to the left of an arrow an even number of times, and *negative* otherwise. It is *strictly positive* if it never occurs to the left of an arrow.

$A \leq B$ in the free ita iff $A \leq B$ can be derived via the following axioms and rule:

- (1) $A \leq B$ if $A = B$ according to the ICS rules.
- (2) $A[D[B \cap C]] \leq A[D[B] \cap D[C]]$ if $D[-]$ is strictly positive
- (3) $A[B \cap C] \leq A[B]$ if $A[-]$ is positive
- (4) $A[B] \leq A[B \cap C]$ if $A[-]$ is negative
- (5) $A \leq B \ \& \ B \leq C \implies A \leq C$

By straightforward induction on derivations, we can show that the free ita validates all of the above rules and that, conversely, postulating these rules to all type expressions built from \rightarrow and \cap results in an ita.

3.5. Set-theoretic. Finally, it is possible to “bake in” the laws of ICS/semilattice by using finite sets directly in our representation language.

Recall that the free semilattice on the set \mathbb{A} is described by the finite powerset of \mathbb{A} , where the union of two finite subsets represents taking the join or meet of the semilattice.

Similarly, to construct a meet semilattice with a left-antitone, right-distributive binary operation \Rightarrow , it suffices to interleave taking finite subsets with introducing new elements built with \Rightarrow . Right-distributivity implies that, for any sets X and Y , we should have

$$X \Rightarrow Y = \{X \Rightarrow y \mid y \in Y\}$$

The elements of the free ita on the set \mathbb{A} can therefore be represented by finitely branching trees, defined inductively by the following rule (the base case being obtained at $k = 0$):

$$\frac{X_1 \subseteq_f \mathcal{A}(\mathbb{A}) \quad \cdots \quad X_k \subseteq_f \mathcal{A}(\mathbb{A}) \quad \alpha \in \mathbb{A}}{X_1 \Rightarrow \cdots \Rightarrow X_k \Rightarrow \alpha \in \mathcal{A}(\mathbb{A})}$$

As we see, this definition naturally makes a distinction between finite subsets of $\mathcal{A}(\mathbb{A})$ — representing intersection types — and the elements themselves, representing arrow and atomic types. This distinction of course reflects the same situation that we encountered with distributivity normal forms in (3.5) and (3.6).

The partial order relation on $\mathcal{A}(\mathbb{A})$ can likewise be defined inductively, following the generation of the elements of $\mathcal{A}(\mathbb{A})$ themselves:

$$\frac{X, Y \subseteq_f \mathcal{A}(\mathbb{A}) \quad \forall y \in Y \exists x \in X. x \leq y}{X \leq Y} \quad \frac{X_1 \leq Y_1 \quad \cdots \quad X_k \leq Y_k \quad \alpha \in \mathbb{A}}{Y_1 \Rightarrow \cdots \Rightarrow Y_k \Rightarrow \alpha \leq X_1 \Rightarrow \cdots \Rightarrow X_k \Rightarrow \alpha}$$

Remark 3.1. The above expressions for $\mathcal{A}(\mathbb{A})$ do not yet give unique representatives with respect to the relation $(\sim) = (\leq \cap \geq)$, because some elements x of a set $X \subseteq_f \mathcal{A}(\mathbb{A})$ can be *redundant*, in the sense that $x \geq \bigcap \{y \in X \mid y \neq x\}$. Then $X \sim X'$, where $X' = X - \{x\}$. This will also produce elements $X \rightarrow \alpha \sim X' \rightarrow \alpha$.

The expressions could be made completely canonical by removing redundant elements hereditarily from X and from all of its subexpressions. This can be done recursively, which therefore yields an effective procedure for computing *the* canonical representative of $[A]_{\sim}$ for every intersection type A . However, we will not need this.

4. THE ESSENTIAL INTERSECTION TYPE ASSIGNMENT SYSTEM

4.1. The original CDV type system. The two-layer grammar of types encountered in the previous section is in fact much closer in spirit to the grammar used in the original [5] paper. Accordingly, the rules of type assignment in that system made a distinction between types and sets/intersections (there called *sequences*). The partial order on the types and the sequences is also present in that paper, if somewhat implicitly.

The original formulation made it possible to characterize solvable terms using intersection types. This system however is not the optimal choice for our purposes, and an even more minimal formulation has been proposed by van Bakel.

4.2. van Bakel's Essential Intersection Types. The following type assignment system closely follows [10], with minimal adjustments for consistency.

The system follows a two-layer grammar:

$$\begin{aligned} A \in \mathbb{T} &::= A \mid \mathbb{T}_\cap \rightarrow \mathbb{T} \\ X \in \mathbb{T}_\cap &::= \mathbb{T} \cap \dots \cap \mathbb{T} \end{aligned}$$

When $X = A_1 \cap \dots \cap A_k$, we often write $\cap A_i$ for X . We may also write $B \in X$ to imply that $B = A_i$ for some i .

Every $A \in \mathbb{T}$ can be written as $A = X_1 \rightarrow \dots \rightarrow X_k \rightarrow \alpha$, for some $X_i \in \mathbb{T}_\cap$ and $\alpha \in \mathbb{A}$. This α is called *the principal atom of A*.

A context is a finite function $\Gamma : \mathbb{V} \rightarrow \mathbb{T}_\cap$.

We write $A \in \Gamma(x)$ if $\Gamma(x) = B_1 \cap \dots \cap B_k$ and $A = B_i$ for some i .

The *essential type assignment system* is defined by the following rules.

$$\frac{A \in \Gamma(x)}{\Gamma \vdash x : A} \quad \frac{\Gamma, x : X \vdash M : B}{\Gamma \vdash \lambda x. M : X \rightarrow B} \quad \frac{\Gamma \vdash M : X \rightarrow B \quad \{\Gamma \vdash N : A\}_{A \in X}}{\Gamma \vdash MN : B}$$

Among the most attractive features of this systems are:

- All types are in distributive normal form.
- (Implicit) subsumption is only allowed on variables.

While these are serious restrictions, they do not change the set of typable terms:

Theorem 4.1. $\Gamma \vdash_{\text{CDV}} M : A$ iff $\Gamma \vdash M : A'$ in the essential system, where $A' \sim A$.

Proof. See Theorems 4.3–4.5 in [10] and the remark that follows. \square

Since the system is completely syntax-directed, the following lemma is immediate. See also [2, 14.1.9].

Lemma 4.2 (Inversion).

- (1) $\Gamma \vdash x : A \iff A \in \Gamma(x)$
- (2) $\Gamma \vdash MN : A \iff \exists X = \cap B_i. \Gamma \vdash M : (X \rightarrow A) \ \& \ \forall i. \Gamma \vdash N : B_i$
- (3) $\Gamma \vdash \lambda x. M : A \iff A = (X \rightarrow B) \ \& \ \Gamma, x : X \vdash M : B$

Definition 4.3. Given Γ and Δ , put $\Gamma \uplus \Delta(x) = \{A \mid A \in \Gamma(x)\} \cup \{A \mid A \in \Delta(x)\}$.

Lemma 4.4 (Thinning). *Let Γ be a context, $A, B \in \mathbb{T}$. Let M be a beta normal form. Suppose that the principal atom of B occurs neither in Γ nor in A . Then*

$$\Gamma \uplus y : B \vdash M : A \implies \Gamma \vdash M : A$$

Proof. Recall that the set of beta normal forms can be characterized by the following grammar:

$$\mathcal{N}_\beta ::= x\mathcal{N}_\beta \cdots \mathcal{N}_\beta \mid \lambda x. \mathcal{N}_\beta \quad (4.1)$$

We proceed by induction on the generation of M according to this grammar.

$M = xM_1 \cdots M_k$: By applying the Inversion Lemma k times, we find $\{X_i\}_{1 \leq i \leq k}$ so that

$$\begin{aligned} X_i &= B_{i,1} \cap \cdots \cap B_{i,k(i)} \\ \Gamma, y : B \vdash x : X_1 \rightarrow \cdots \rightarrow X_k \rightarrow A \\ (\forall i \forall j) \quad \Gamma, y : B \vdash M_i : B_{i,j} \end{aligned}$$

By inversion, $x : X_1 \rightarrow \cdots \rightarrow X_k \rightarrow A \in (\Gamma, y : B)(x)$.

But since the principal atom of B does not occur in A , we must actually have $X_1 \rightarrow \cdots \rightarrow X_k \rightarrow A \in \Gamma(x)$. In particular

$$\Gamma \vdash x : X_1 \rightarrow \cdots \rightarrow X_k \rightarrow A$$

Moreover, we see that X_i occurs in Γ and hence $B_{i,j}$ occurs in Γ , for all i and j . By induction hypothesis, we thus also have

$$(\forall i \forall j) \quad \Gamma \vdash M_i : B_{i,j}$$

By the application rule, $\Gamma \vdash x\vec{M} : A$.

$M = \lambda x. M'$: By inversion, we must have $A = X \rightarrow C$, where

$$\Gamma, y : B, x : X \vdash M' : C$$

Since X and C are both subexpressions of A , the principal atom of B does not occur in them either.

So the induction hypothesis applies directly, and we get

$$\Gamma, x : X \vdash M' : C$$

By the abstraction rule, $\Gamma \vdash \lambda x. M' : X \rightarrow C$.

That is, $\Gamma \vdash M : A$. □

Corollary 4.5. *Let Γ and Δ be contexts and suppose that $\Gamma \uplus \Delta \vdash M : A$.*

If the type atoms of Δ occur neither in Γ nor in A , then $\Gamma \vdash M : A$.

Proof. By induction on Δ , using the previous lemma. □

5. UNIQUENESS TYPING

Definition 5.1. Let $M \in \Lambda$. A *uniqueness typing* for M is a pair (Γ, A) such that

- (1) $\Gamma \vdash M : A$
- (2) $\Gamma \vdash N : A \implies M =_{\beta\eta} N$

In this section, we will show that every strongly normalizing term admits a uniqueness typing in CDV.

- Notation 5.2.** (1) For $M \in \Lambda$, $\text{NF}_{\beta(\eta)}(M)$ is the $\beta(\eta)$ -normal form of M (if it exists).
 (2) \mathcal{N}_β is the set of all beta normal forms.
 (3) $\mathcal{N}_{\beta\eta}$ is the set of all beta+eta normal forms.
 (4) \mathcal{SN} is the set of all strongly normalizing terms.

We will now establish the following progression of claims.

Proposition 5.3. *For every $M \in \mathcal{N}_\beta$ there exists a context Γ and an intersection type A such that*

- (1) $\Gamma \vdash M : A$
- (2) $\forall N \in \mathcal{N}_\beta. \Gamma \vdash N : A \implies M \twoheadrightarrow_\eta N$

Corollary 5.4. *For every $M \in \mathcal{N}_\beta$ there exists a context Γ and an intersection type A such that*

- (1) $\Gamma \vdash M : A$
- (2) $\forall N \in \Lambda. \Gamma \vdash N : A \implies \text{NF}_{\beta\eta}(M) \equiv \text{NF}_{\beta\eta}(N)$

Theorem 5.5. *For every $M \in \mathcal{SN}$ there exists a context Γ and an intersection type A such that*

- (1) $\Gamma \vdash M : A$
- (2) $\forall N \in \Lambda. \Gamma \vdash N : A \implies M =_{\beta\eta} N$

Proof of Proposition 5.3. Let M be given.

For every position p in M , let α_p be a fresh type atom.

We shall again make use of the following grammar for beta normal forms:

$$\mathcal{N}_\beta ::= x\mathcal{N}_\beta \cdots \mathcal{N}_\beta \mid \lambda x. \mathcal{N}_\beta \quad (5.1)$$

We proceed by induction on the generation of M according to this grammar.

Case 1. $M = xM_1 \cdots M_k, k \geq 0$.: (This includes the base case $M = x$.)

By induction hypothesis, there exist $\Gamma_1, \dots, \Gamma_k, A_1, \dots, A_k$ such that

- (1) $\Gamma_i \vdash M_i : A_i$
- (2) $\forall N \in \mathcal{N}_\beta. \Gamma_i \vdash N : A_i \implies M_i \twoheadrightarrow_\eta N$

Let $\alpha = \alpha_p$ be the unique type atom associated to the current subterm. Put

$$\begin{aligned} \Gamma &= \Gamma_1 \uplus \cdots \uplus \Gamma_k \uplus \{x : A_1 \rightarrow \cdots \rightarrow A_k \rightarrow \alpha\} \\ A &= \alpha \end{aligned}$$

Since Γ extends each Γ_i , by weakening we have $\Gamma \vdash M_i : A_i$

By k uses of the application rule, we obtain $\Gamma \vdash x\vec{M} : \alpha$. That is, $\Gamma \vdash M : A$.

Now let $N \in \mathcal{N}_\beta$ and suppose $\Gamma \vdash N : A$.

We consider the possible shapes of N according to (5.1).

If $N = \lambda y. N'$, then applying the inversion lemma to $\Gamma \vdash N : A$ yields that A must be a function type $X \rightarrow A'$, contradicting that $A = \alpha$ is an atom.

Thus N is an application: $N = yN_1 \cdots N_l$.

Applying inversion to $\Gamma \vdash N : A$ a sufficient number of times now yields that

$$\begin{aligned} \Gamma \vdash y : Y_1 \rightarrow \cdots \rightarrow Y_l \rightarrow A \\ \Gamma \vdash N_j : B \quad (1 \leq j \leq l, B \in Y_j) \end{aligned}$$

However, the only element in the context Γ which contains the atom $A = \alpha$ is

$$A_1 \rightarrow \cdots \rightarrow A_k \rightarrow \alpha \in \Gamma(x)$$

So we must have $y = x$, $l = k$, $N = xN_1 \cdots N_k$, and $Y_i = A_i$ for all i . That is,

$$\Gamma \vdash N_i : A_i$$

By applying Lemma 4.4 to this judgment with $y : B$ being $x : A_1 \rightarrow \cdots \rightarrow A_k \rightarrow \alpha$, followed by Corollary 4.5 with Δ being $\uplus_{j \neq i} \Gamma_j$, we find $\Gamma_i \vdash N_i : A_i$.

By induction hypothesis, we have $M_i \twoheadrightarrow_\eta N_i$.

Now $M = xM_1 \cdots M_k \twoheadrightarrow_\eta xN_1 \cdots N_k = N$, completing the proof of this case.

Case 2. $M = \lambda x.M'.$: By induction hypothesis, there exist Γ', A' such that

- (1) $\Gamma' \vdash M' : A'$
- (2) $\forall N' \in \mathcal{N}_\beta. \Gamma' \vdash N' : A' \implies M' \twoheadrightarrow_\eta N'$

If x occurs in M' , then by the free variable lemma, $\Gamma'(x)$ is defined.

Otherwise, in the definition below, set $X = \beta$, a fresh atom.

$$\begin{aligned} X &= \Gamma'(x) = B_1 \cap \cdots \cap B_k \\ A &= X \rightarrow A' \\ \Gamma &= \Gamma' - \{x : X\} \end{aligned}$$

By the typing rule for abstraction, we get $\Gamma \vdash M : A$.

Now let $N \in \mathcal{N}_\beta$ satisfy $\Gamma \vdash N : A$. We have two cases.

Case 2.1: $N = \lambda y.N'.$: Then $\Gamma \vdash \lambda y.N' : X \rightarrow A'$.

By inversion, we get $\Gamma, y : X \vdash N' : A'$.

In other words, $\Gamma, x : X \vdash N'[y := x] : A'$.

That is, $\Gamma' \vdash N'[y := x] : A'$.

Now part 2 of IH yields that $M' \twoheadrightarrow_\eta N'[y := x]$, hence

$$M = \lambda x.M' \twoheadrightarrow_\eta \lambda x.N'[y := x] =_\alpha \lambda y.N' = N$$

Case 2.2: $N = yN_1 \cdots N_l.$: Then $\Gamma' - \{x : X\} \vdash yN_1 \cdots N_l : X \rightarrow A'$.

By the free variable lemma, $x \notin \text{FV}(N)$.

By weakening, we also have $\Gamma' \vdash N : X \rightarrow A'$.

Since $\Gamma' \vdash x : X$, application yields $\Gamma' \vdash Nx : A'$.

By part 2 of IH, we get $M' \twoheadrightarrow_\eta Nx$.

Hence $M = \lambda x.M' \twoheadrightarrow_\eta \lambda x.Nx \twoheadrightarrow_\eta N$, where the last step uses $x \notin \text{FV}(N)$.

This concludes the proof of the statement. \square

Proof of Corollary 5.4. Let $M \in \mathcal{N}_\beta$ be given.

By Proposition 5.3, let (Γ, A) be such that

$$\forall N \in \mathcal{N}_\beta. \quad \Gamma \vdash N : A \implies M \twoheadrightarrow_\eta N \quad (5.2)$$

Let $N \in \Lambda$, and suppose $\Gamma \vdash N : A$. By the Characterization Theorem of strongly normalizing terms [BDS, 17.2.15(iii)], N is strongly normalizing. So $N \twoheadrightarrow_\beta \mathbf{NF}_\beta(N)$.

By Subject Reduction for beta [BDS, 14.2.3], $\Gamma \vdash \mathbf{NF}_\beta(N) : A$.

Since η -reduction is SN, we also have $\mathbf{NF}_\beta(N) \twoheadrightarrow_\eta \mathbf{NF}_{\beta\eta}(N)$.

By Subject Reduction for eta [BDS, 14.2.8(i)], $\Gamma \vdash \mathbf{NF}_{\beta\eta}(N) : A$.

By (5.2), $M \twoheadrightarrow_\eta \mathbf{NF}_{\beta\eta}(N)$. Hence $\mathbf{NF}_{\beta\eta}(M) \equiv \mathbf{NF}_{\beta\eta}(N)$. \square

Proof of Theorem 5.5. Let $M \in \mathcal{SN}$.

We proceed by induction on the *longest* reduction $M \twoheadrightarrow_\beta \mathbf{NF}_\beta(M)$.

Case 1: $M \equiv \mathbf{NF}_\beta(M)$.: Immediate by Corollary 5.4.

Case 2: $M \equiv C[(\lambda x.P)Q] \rightarrow C[P[x := Q]] \twoheadrightarrow \mathbf{NF}_\beta(M), x \in \mathbf{FV}(P)$.:

Let $M' = C[P[x := Q]]$.

By induction hypothesis, let (Γ, A) be such that

- (1) $\Gamma \vdash M' : A$
- (2) $\forall N \in \Lambda. \Gamma \vdash N : A \implies M' =_{\beta\eta} N$

Since $x \in \mathbf{FV}(P)$, $(\lambda x.P)Q$ is a $\lambda\mathbf{I}$ -redex, and therefore validates the subject expansion property [BDS, 14.2.5(i)].

Thus $\Gamma \vdash M : A$ as well, which completes this case since $M =_\beta M'$.

Case 3: $M \equiv C[(\lambda x.P)Q] \rightarrow C[P[x := Q]] \twoheadrightarrow \mathbf{NF}_\beta(M), x \notin \mathbf{FV}(P)$.:

Let $M' = C[P[x := Q]] = C[P]$.

By induction hypothesis, let (Γ, A) be such that

- (1) $\Gamma \vdash M' : A$
- (2) $\forall N \in \Lambda. \Gamma \vdash N : A \implies M' =_{\beta\eta} N$

Note that $Q \in \mathcal{N}_\beta$, for otherwise the redex $(\lambda x.P)Q$ would not be contracted in the longest reduction $M \twoheadrightarrow_\beta \mathbf{NF}_\beta(M)$. (A longer reduction could be obtained by first contracting redexes still present in Q .)

By Proposition 5.3, let (Δ, B) be such that

- (1) $\Delta \vdash Q : B$
- (2) $\forall V \in \mathcal{N}_\beta. \Delta \vdash V : B \implies Q \twoheadrightarrow_\eta V$

Without loss of generality, we may assume that the type atoms of Δ and B are disjoint from those of Γ and A .

By weakening, we have $\Gamma \uplus \Delta \vdash M' : A$.

Hence, we can also get $\Gamma \uplus \Delta \vdash M : A$, by replacing the subderivation δ for the subterm P as follows:

$$\frac{\frac{\delta}{P : \psi}}{\lambda x.P : B \rightarrow \psi} \quad Q : B \quad \frac{}{(\lambda x.P)Q : \psi}$$

Let N be given, and suppose $\Gamma \uplus \Delta \vdash N : A$.

By subject reduction, $\Gamma \uplus \Delta \vdash \mathbf{NF}_{\beta\eta}(N) : A$.

By the thinning lemma, $\Gamma \vdash \mathbf{NF}_{\beta\eta}(N) : A$.

By hypothesis 2 on (Γ, A) we obtain $M \rightarrow M' =_{\beta\eta} N$. \square

6. CONCLUSION

We have shown that intersection types admit typings that are dual in spirit to principal typings. Whereas principal typings are the most conservative, assuming the minimum needed to type a given term, uniqueness typings assume as much as is needed to ensure that the given term is, up to beta-eta equality, the only term of that type.

REFERENCES

- [1] Roberto M. Amadio and Pierre-Louis Curien. *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.
- [2] H. Barendregt, W. Dekkers, and R. Statman. *Lambda Calculus with Types*. Perspectives in Logic. Cambridge University Press, 2013.
- [3] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. Symb. Log.*, 48(4):931–940, 1983.
- [4] Carraro, Alberto and Salibra, Antonino. Easy lambda-terms are not always simple. *RAIRO-Theor. Inf. Appl.*, 46(2):291–314, 2012.
- [5] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Mathematical Logic Quarterly*, 27(2-6):45–58, 1981.
- [6] M. Dezani-Ciancaglini and S. Ghilezan. Lambda models characterizing computational behaviours of terms. *Schedae Informaticae*, 12:35–49, June 2003.
- [7] Delia Kesner and Pierre Vial. Types as resources for classical natural deduction. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, volume 84 of *LIPIcs*, pages 24:1–24:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
- [8] Simona Ronchi Della Rocca. Intersection Types and Denotational Semantics: An Extended Abstract (Invited Paper). In Silvia Ghilezan, Herman Geuvers, and Jelena Ivetić, editors, *22nd International Conference on Types for Proofs and Programs (TYPES 2016)*, volume 97 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:7, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [9] Rick Statman. A finite model property for intersection types. In Jakob Rehof, editor, *Proceedings Seventh Workshop on Intersection Types and Related Systems, ITRS 2014, Vienna, Austria, 18 July 2014*, volume 177 of *EPTCS*, pages 1–9, 2014.
- [10] Steffen van Bakel. Intersection type assignment systems. *Theoretical Computer Science*, 151(2):385–435, 1995. 13th Conference on Foundations of Software Technology and Theoretical Computer Science.