Toward Isomorphism of Intersection and Union Types*

Mario Coppo
Dip. di Informatica, Univ. di Torino, Italy

Mariangiola Dezani-Ciancaglini
Dip. di Informatica, Univ. di Torino, Italy

Ines Margaria
Dip. di Informatica, Univ. di Torino, Italy

Maddalena Zacchi
Dip. di Informatica, Univ. di Torino, Italy

In this paper we investigate type isomorphism in a λ-calculus with intersection and union types. It is known that in λ-calculus, the isomorphism between two types is realised by an invertible term. Notably all invertible terms are linear terms. Type isomorphism is usually proved using a form of Inversion Lemma to relate terms and types. Currently in the literature there is no Inversion Lemma for intersection and union types. Moreover, the subject reduction property doesn't hold.

In the present paper we prove an Inversion Lemma for linear terms in the intersection and union type system, showing also that subject reduction holds for them. Invertible terms being linear terms, this inversion lemma gives a tool to investigate isomorphism of intersection and union types. We define a sufficient condition to assure this isomorphism. We conjecture that this condition is also necessary. The validity of the conjecture would provide a complete characterisation of type isomorphism.

1 Introduction

In a calculus with types, two types σ and τ are called *isomorphic* if there exist two terms P of type $\sigma \to \tau$ and P' of type $\tau \to \sigma$ such that both their compositions $P \circ P'$ and $P' \circ P$ give the identity (at the proper type). The study of type isomorphism started in the 1980 with the aim to find all the type isomorphisms valid in every model of a given language [3]. If one looks at this problem choosing as language a λ-calculus with types, one can immediately note the close relation between the type isomorphism and the λ-term invertibility. Actually in the untyped λ-calculus a λ-term P is *invertible* if there is a λ-term P' such that $P \circ P' = P' \circ P = \mathbf{I}$ ($\mathbf{I} \equiv \lambda x.x$). The problem of term invertibility has been extensively studied for the untyped λ-calculus since 1970 and the main result has been the complete characterisation of the invertible λ-terms in λβη-calculus [4]: the invertible terms are all and only the *finite hereditary permutators*.

Definition 1.1 (Finite Hereditary Permutators)**.** *A* finite hereditary permutator *(f.h.p. for short) is a* λ*-term of the form (modulo* βη*-conversion)*

$$\lambda x y_1 \ldots y_n . x(P_1 y_{\pi(1)}) \ldots (P_n y_{\pi(n)}) \quad (n \geq 0)$$

where π *is a permutation of* $1, \ldots, n$*, and* $P_1 \ldots P_n$ *are f.h.p.*

Note that the identity $\lambda x.x$ is trivially a f.h.p. (take $n = 0$). Another example of f.h.p. is $\lambda x y_1 y_2 . x y_2 y_1 = \lambda x y_1 y_2 . x((\lambda z.z) y_2)((\lambda z.z) y_1)$.

Theorem 1.2. *A* λ*-term is invertible iff it is a finite hereditary permutator.*

This result, obtained in the framework of the untyped λ-calculus, has been the basis for studying type isomorphism in different type systems for the λ -calculus. Note that the f.h.p.s have closed λ-normal forms; moreover every f.h.p. P has, modulo βη-conversion, a unique inverse P^{-1}. Taking into

*The first three authors are supported by the MIUR Project IPODS. The last author is supported by the MIUR Project BRR.

account these properties, the definition of type isomorphism in a λ-calculus with types can be stated as follows:

Definition 1.3 (Type isomorphism)**.** *Given a λ-calculus with types, two types σ and τ are isomorphic ($\sigma \approx \tau$) if there exists a f.h.p. P such that $\vdash P : \sigma \to \tau$ and $\vdash P^{-1} : \tau \to \sigma$. In this case we say that P proves the isomorphism.*

The main line used to characterise the isomorphisms in a given type system has been to provide a suitable set of equations and to prove that these equations induce the type isomorphism w.r.t. βη-conversion, i.e. that the types of the f.h.p.s are all and only those induced by the set of equations.

The first typed λ-calculus studied has been the simply typed λ-calculus, for which it is proved [3] that it is necessary a unique equation, the **swap** equation:

$$\sigma \to \tau \to \rho = \tau \to \sigma \to \rho$$

Afterwords the study has been directed toward richer λ-calculi, obtained from the simply typed λ-calculus by adding, in an incremental way, some other type constructors (like product and unit types) or by allowing higher order types (System F). The equations characterising the type isomorphism are summarised in [6]; one can note that also the sets of equations obey to an incremental law in the sense that the set of equations for a typed λ-calculus obtained by adding a primitive to a given λ-calculus, results to be an extension of the set of equations of the λ-calculus without that primitive.

In the presence of intersection types this incremental approach doesn't work as pointed out in [5]; in particular with intersection types, the isomorphism is no longer a congruence and type equality in the standard models of intersection types doesn't entail type isomorphism. These quite unexpected facts required the introduction of a syntactical notion of type similarity in order to fully characterise the isomorphic types [5].

The study of isomorphism looks even harder for type systems with intersection and union types because for these systems, in general, the subject reduction property doesn't hold and no kind of *Inversion Lemma* has been provided so far (see [2]). This is a real technical difficulty because the Inversion Lemma allows a reverse reading of the inference rules. In particular if the subject of the conclusion is an application or an abstraction, then it gives some information on types of its immediate subterms.

In this paper we will show that in a type system with intersection and union types both subject reduction and an Inversion Lemma hold for *linear* terms, i.e. terms in which each variable occurs exactly once. Since the f.h.p.s are linear terms this provides the tool to investigate type isomorphism. We will then prove some interesting isomorphisms corresponding to basic properties of set and function theories. We will finally define a notion of similarity for intersection and union types that implies isomorphism and, we conjecture, fully characterises it.

2 Subject reduction for linear terms

The union/intersection type system considered in this paper is the basic one introduced in the seminal paper [7]. We only omit to include the universal type ω (as in [5]), so considering only typeable terms which have a normal form and cannot be equated in any theory.

The formal syntax of intersection and union types is:

$$\sigma \quad := \quad \varphi \;\mid\; \sigma \to \sigma \;\mid\; \sigma \wedge \sigma \;\mid\; \sigma \vee \sigma$$

where φ denotes an atomic type. Let $\sigma, \tau, \rho, \theta$ range over types. Types are considered modulo idempotence, commutativity and associativity of $\wedge$ and $\vee$. We are then allowed to write $\bigwedge_{i \in I} \sigma_i$ and $\bigvee_{i \in I} \sigma_i$ with

$$(Ax) \qquad \Gamma, x:\sigma \vdash x:\sigma$$

$$(\to I) \quad \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x.M:\sigma\to\tau} \qquad (\to E) \quad \frac{\Gamma \vdash M:\sigma\to\tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash MN:\tau}$$

$$(\wedge I) \quad \frac{\Gamma \vdash M:\sigma \;\; \Gamma \vdash M:\tau}{\Gamma \vdash M:\sigma\wedge\tau} \qquad (\wedge E) \quad \frac{\Gamma \vdash M:\sigma\wedge\tau}{\Gamma \vdash M:\sigma} \qquad (\vee I) \quad \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash M:\sigma\vee\tau}$$

$$(\vee E) \quad \frac{\Gamma, x:\sigma \vdash M:\rho \quad \Gamma, x:\tau \vdash M:\rho \quad \Gamma \vdash N:\sigma\vee\tau}{\Gamma \vdash M[N/x]:\rho}$$

Figure 1: Typing Rules

finite I. Conventionally, we omit parentheses according to the precedence rule "$\vee$ and $\wedge$ over $\to$" and we assume that $\to$ associates to the right.

We will introduce two type assignment systems for these types, the standard one introduced in [2] and another one with a slightly more general version of the $\vee$ elimination rule. The latter version has the same properties of the former one, but allows to prove more types isomorphisms. Since the Inversion Lemma and subject reduction for linear terms have an interest in themselves, in this section we will state them for the standard system. The extensions to the second system are in most cases straightforward (one minor difference will be explicitly remarked).

The standard type assignment system is the system of intersection and union types for the ordinary λ-calculus with the typing rules of Fig. 1. As usual the environment Γ contains only one statement for any variable.

It is easy to verify that the following rule:

$$(\wedge L) \quad \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma, x:\sigma\wedge\rho \vdash M:\tau}$$

is admissible.

As remarked in the introduction, in this system types are not preserved by β-reduction. In fact, because of rule $(\vee E)$, the correspondence between subterms and subdeductions is lost, and so it is sometimes impossible to reconstruct a type deduction for the reduct of a typed term. This is also the reason why, in general, no kind of Inversion Lemma has been proved for this system.

For example, one can deduce the type: $(\sigma\to\sigma\to\tau)\wedge(\rho\to\rho\to\tau)\to(\theta\to\sigma\vee\rho)\to\theta\to\tau$ both for $\lambda xyz.x(yz)(yz)$ and $\lambda xyz.x(\mathbf{I}yz)(\mathbf{I}yz)$, but this type can be deduced neither for $\lambda xyz.x(\mathbf{I}yz)(yz)$ nor for $\lambda xyz.x(yz)(\mathbf{I}yz)$.

To overcome this difficulty two solutions have been proposed in [2]: the introduction of a parallel reduction strategy and the use of type theories to enrich the assignment system axiomatizing inequalities representing semantic inclusion of types. Both techniques are not interesting here since we want keep the original introduction and elimination rules and work with standard $\beta\eta$-reduction. It is also known [1] that, when union is considered, types are preserved for call-by-value λ-calculus [8], obtained by restricting the β-rule to redexes whose argument is a value, i.e. a variable or an abstraction, but this also requires to be bound to a specific reduction strategy.

In the following we prove that both the subject reduction property and an Inversion Lemma hold in the basic system if we restrict ourself to consider linear terms. Note, for instance, that the term used

above as counter example of subject reduction is not linear and that the problem arises when only one of the two instances of $\mathbf{I}yz$ is reduced.

Since finite hereditary permutators are linear terms this restriction is not relevant as far as we are interested in investigating type isomorphisms.

Before approaching the main results of this section, let us state a lemma proving a property of type derivations for abstractions, which will be used in the proof of the Inversion Lemma.

Lemma 2.1. *If* Π *is a derivation of* $\Gamma \vdash \lambda x.M : \bigwedge_{i\in I}\sigma_i$ *then for each* $i \in I$ *there exists a derivation* Π_i *of* $\Gamma \vdash \lambda x.M : \sigma_i$ *such that the number of applied rules in* Π_i *is less than that in* Π.

Proof. The proof is by induction on derivations. □

Note that an analogous lemma for the $\vee$-constructor doesn't hold. In fact it is easy to prove that

$$z : \sigma\vee\tau \vdash \lambda x.xz : ((\sigma\to\sigma)\wedge(\tau\to\tau)\to\sigma)\vee((\sigma\to\sigma)\wedge(\tau\to\tau)\to\tau)$$

but $z : \sigma\vee\tau \nvdash \lambda x.xz : (\sigma\to\sigma)\wedge(\tau\to\tau)\to\sigma$ and $z : \sigma\vee\tau \nvdash \lambda x.xz : (\sigma\to\sigma)\wedge(\tau\to\tau)\to\tau$.

To state the *Inversion Lemma* for linear terms it is useful to introduce the following relations on the set of types. The relation "fit", denoted by $\triangledown$, permits to link the types deduced for a variable to the assumption in the environment for the variable itself. The relations "agree", denoted by $\rhd$, and "co-agree", denoted by $\lhd$, relate types of applications and abstractions to types of their subterms.

Definition 2.2. *Define* $\sigma\triangledown\tau$ *as the minimal reflexive and transitive relation satisfying:*

$$\begin{array}{ccc}
\sigma\triangledown\tau \textit{ and } \sigma\triangledown\rho & \textit{imply} & \sigma\triangledown\tau\wedge\rho \\
\sigma\triangledown\tau\wedge\rho & \textit{implies} & \sigma\triangledown\tau \\
\sigma\triangledown\tau & \textit{implies} & \sigma\triangledown\tau\vee\rho \\
\sigma\triangledown\rho \textit{ and } \tau\triangledown\rho & \textit{imply} & \sigma\vee\tau\triangledown\rho.
\end{array}$$

Definition 2.3. *Define* $\sigma\rhd\tau\to\rho$ *by:*

$$\begin{array}{ccc}
\sigma\to\tau\rhd\sigma\to\tau & & \\
\sigma_1\rhd\tau_1\to\rho_1 \textit{ and } \sigma_2\rhd\tau_2\to\rho_2 & \textit{imply} & \sigma_1\wedge\sigma_2\rhd\tau_1\wedge\tau_2\to\rho_1\wedge\rho_2 \\
\sigma\rhd\tau\to\rho_1\wedge\rho_2 & \textit{implies} & \sigma\rhd\tau\to\rho_1 \\
\sigma\rhd\tau\to\rho & \textit{implies} & \sigma\rhd\tau\to\rho\vee\theta \\
\sigma_1\rhd\tau_1\to\rho \textit{ and } \sigma_2\rhd\tau_2\to\rho & \textit{imply} & \sigma_1\wedge\sigma_2\rhd\tau_1\vee\tau_2\to\rho \textit{ and} \\
 & & \sigma_1\vee\sigma_2\rhd\tau_1\wedge\tau_2\to\rho.
\end{array}$$

Definition 2.4. *Define* $\sigma\lhd\tau\to\rho$ *by:*

$$\begin{array}{ccc}
\sigma\to\tau\lhd\sigma\to\tau & & \\
\sigma\lhd\tau\to\rho & \textit{implies} & \sigma\vee\theta\lhd\tau\to\rho \\
\sigma_1\lhd\tau_1\to\rho_1 \textit{ and } \sigma_2\lhd\tau_2\to\rho_2 & \textit{imply} & \sigma_1\wedge\sigma_2\lhd\tau_1\wedge\tau_2\to\rho_1\vee\rho_2.
\end{array}$$

The introduced relations are used to state an Inversion Lemma for linear terms and a Key Lemma, from which the subject reduction property for linear terms easily follows. The former characterizes the types of the immediate subterms of a given term and the latter says that the types that the system allows to assign to terms which are abstractions go along (co-agree) with arrow types.

Lemma 2.5 (Inversion Lemma for *linear* terms)**.** 1. *If* $\Gamma, x:\sigma \vdash x:\tau$, *then* $\sigma \triangledown \tau$.

2. *If* $\Gamma \vdash MN:\sigma$ *and* MN *is a linear term, then there are* τ,ρ *such that* $\Gamma \vdash M:\tau$, $\Gamma \vdash N:\rho$ *and* $\tau \rhd \rho \to \sigma$.

3. *If* $\Gamma \vdash \lambda x.M:\sigma$ *and* M *is a linear term, then there are* τ,ρ *such that* $\Gamma, x:\tau \vdash M:\rho$ *and* $\sigma \lhd \tau \to \rho$.

Proof. By induction on derivations. □

Lemma 2.6 (Key lemma for SR)**.** *If* $\Gamma \vdash \lambda x.M:\sigma$ *and* $\sigma \rhd \tau \to \rho$, *then* $\Gamma, x:\tau \vdash M:\rho$.

Proof. By induction on derivations. □

Theorem 2.7 (SR for *linear* terms)**.** *If* M *is a linear term and* $\Gamma \vdash M:\sigma$ *and* $M \longrightarrow^* N$, *then* $\Gamma \vdash N:\sigma$.

Proof. It is easy to show that $\Gamma \vdash (\lambda x.M)N:\sigma$ implies $\Gamma \vdash M[N/x]:\sigma$ using the Inversion and Key Lemmas. □

3 Type isomorphism

The following basic isomorphisms are directly related to standard properties of functional types. The η-expansion of the identity $\lambda xy.xy$ shows them.

Lemma 3.1. *The following isomorphisms hold:*

$$\to \wedge\textbf{comm}.\quad \sigma \to \tau \wedge \rho \approx (\sigma \to \tau) \wedge (\sigma \to \rho)$$
$$\to \vee\textbf{comm}.\quad \sigma \vee \tau \to \rho \approx (\sigma \to \rho) \wedge (\tau \to \rho)$$

To prove the isomorphism of the distributive laws of set theory we need instead to enforce the $(\vee E)$ rule in the following way:

$$(\vee E')\quad \frac{\Gamma, x:\sigma \wedge \theta \vdash M:\rho \quad \Gamma, x:\tau \wedge \theta \vdash M:\rho \quad \Gamma \vdash N:(\sigma \vee \tau) \wedge \theta}{\Gamma \vdash M[N/x]:\rho}$$

Note that rule $(\vee E)$ is admissible in the system with rule $(\vee E')$ by taking $\theta = \sigma \vee \tau$, since we can derive $x:\sigma \wedge (\sigma \vee \tau)$ from $x:\sigma$ and similarly for τ.

Using rule $(\vee E')$ and again the identity we can prove the following:

Lemma 3.2. *The following isomorphisms hold:*

$$\textbf{Dist}\wedge\vee.\quad \rho \wedge (\sigma \vee \tau) \approx (\rho \wedge \sigma) \vee (\rho \wedge \tau)$$
$$\textbf{Dist}\vee\wedge.\quad \rho \vee (\sigma \wedge \tau) \approx (\rho \vee \sigma) \wedge (\rho \vee \tau)$$

Only **Dist**$\wedge\vee$ left-to-right and **Dist**$\vee\wedge$ right-to-left need rule $(\vee E')$. For the other two directions rule $(\vee E)$ is enough.

It is interesting to remark that all isomorphisms introduced in this section are provable equalities in the system $\mathbf{B}_+$ of relevant logic [9].

It is crucial to remark that the Inversion Lemma for linear terms (Lemma 2.5) holds for the extended system if we replace the last clause in the definition of "fit" (Definition 2.2) by:

$$\sigma \wedge \theta \triangledown \rho \text{ and } \tau \wedge \theta \triangledown \rho \quad \text{imply} \quad (\sigma \vee \tau) \wedge \theta \triangledown \rho.$$

We introduce now a relation of similarity between sequences of types and between types which assures isomorphism of the related types. The basic aim of this relation is that of formalising isomorphism determined by argument permutations. Similarity is the most interesting relation preserving type isomorphism and it will play the fundamental role in our (conjectured) complete characterisation of all provable isomorphisms when the system with rule $(\vee E')$ is considered. In this definition we must be careful to take into account the fact that, for two types to be isomorphic, it is not sufficient that they coincide modulo permutations of types in the arrow sequences, as in the case of cartesian products. The permutation must be the same for all the corresponding type pairs in an intersection or in an union. The notion of similarity exactly expresses such property.

Definition 3.3 (Similarity)**.** *The* similarity *relation between two sequences of types* $\langle \sigma_1, \ldots, \sigma_m \rangle$, $\langle \tau_1, \ldots, \tau_m \rangle$, *written* $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$, *is the smallest equivalence relation such that:*

1. $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \sigma_1, \ldots, \sigma_m \rangle$;
2. *if* $\langle \sigma_1, \ldots, \sigma_i, \sigma_{i+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \tau_{i+1}, \ldots, \tau_m \rangle$, *then*

$$\langle \sigma_1, \ldots, \sigma_i \wedge \sigma_{i+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i \wedge \tau_{i+1}, \ldots, \tau_m \rangle \text{ and}$$
$$\langle \sigma_1, \ldots, \sigma_i \vee \sigma_{i+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i \vee \tau_{i+1}, \ldots, \tau_m \rangle$$

3. *if* $\langle \sigma_i^{(1)}, \ldots, \sigma_i^{(m)} \rangle \sim \langle \tau_i^{(1)}, \ldots, \tau_i^{(m)} \rangle$ *for* $1 \leq i \leq n$ *and* $\langle \rho_1, \ldots, \rho_m \rangle \sim \langle \rho'_1, \ldots, \rho'_m \rangle$, *then*

$$\langle \sigma_1^{(1)} \to \ldots \to \sigma_n^{(1)} \to \rho_1, \ldots, \sigma_1^{(m)} \to \ldots \to \sigma_n^{(m)} \to \rho_m \rangle \sim$$
$$\langle \tau_{\pi(1)}^{(1)} \to \ldots \to \tau_{\pi(n)}^{(1)} \to \rho'_1, \ldots, \tau_{\pi(1)}^{(m)} \to \ldots \to \tau_{\pi(n)}^{(m)} \to \rho'_m \rangle,$$

where π *is a permutation of* $1, \ldots, n$.

Similarity between types *is trivially defined as similarity between unary sequences:* $\sigma \sim \tau$ *if* $\langle \sigma \rangle \sim \langle \tau \rangle$.

As a first example note that we have immediately $\sigma \to \tau \to \rho \sim \tau \to \sigma \to \rho$, which represents the basic argument swapping. The isomorphism is shown by the permutator $\lambda x y_1 y_2 . x y_2 y_1$
As a less simple example the types:
$\sigma = (\varphi_1 \to \varphi_2 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_3 \to \varphi_4 \to \varphi_3) \to ((\varphi_1 \to \varphi_2 \to \varphi_3) \to \varphi_1) \vee ((\varphi_3 \to \varphi_4 \to \varphi_5) \to \varphi_3)$ and
$\tau = (\varphi_2 \to \varphi_1 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_4 \to \varphi_3 \to \varphi_3) \to ((\varphi_2 \to \varphi_1 \to \varphi_3) \to \varphi_1) \vee ((\varphi_4 \to \varphi_3 \to \varphi_5) \to \varphi_3)$
are similar. The f.h.p. showing the isomorphism is $P = \lambda x y_1 y_2 . x (\lambda z_1 z_2 . y_1 z_2 z_1)(\lambda z_1 z_2 . y_2 z_2 z_1)$, where $\vdash P : \sigma \to \tau$ and $\vdash P^{-1} : \tau \to \sigma$. Note that this typing would fail if we replace in τ the subtype
$((\varphi_2 \to \varphi_1 \to \varphi_3) \to \varphi_1) \vee ((\varphi_4 \to \varphi_3 \to \varphi_5) \to \varphi_3)$ with
$((\varphi_1 \to \varphi_2 \to \varphi_3) \to \varphi_1) \vee ((\varphi_4 \to \varphi_3 \to \varphi_5) \to \varphi_3)$, since the arguments are permuted in only one of the branches of the $\vee$ operator.

Another example is:
$\sigma = (\varphi_1 \to \varphi_2 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_3 \to \varphi_4 \to \varphi_3) \to (\varphi_1 \to \varphi_2 \to \varphi_3 \to \varphi_1) \vee (\varphi_3 \to \varphi_4 \to \varphi_5 \to \varphi_3)$ and
$\tau = (\varphi_2 \to \varphi_1 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_4 \to \varphi_3 \to \varphi_3) \to (\varphi_3 \to \varphi_2 \to \varphi_1 \to \varphi_1) \vee (\varphi_5 \to \varphi_4 \to \varphi_3 \to \varphi_3)$. The f.h.p. showing the isomorphism is $P = \lambda x y_1 y_2 y_3 y_4 . x(\lambda z_1 z_2 . y_1 z_2 z_1) y_4 y_3 y_2$.

The following Lemma is proved by induction on the definition of $\sim$.

Lemma 3.4. *If* $\sigma \sim \tau$ *then* $\sigma \approx \tau$.

We conjecture that this lemma is true also in the opposite direction for types in a normal form to be defined, thus giving a complete characterisation of the isomorphisms provable in the λ-calculus with intersection and union types.

3.1 Normal Forms and the completeness conjecture

The notion of similarity is not enough to characterise all possible isomorphisms. We need (at least) to exploit some provable inclusion properties of types and the basic laws introduced above (i.e. $\to \wedge$**comm**, $\to \vee$**comm**, **Dist**$\wedge\vee$, **Dist**$\vee\wedge$**)**, allowing to apply them (when possible) also at the level of subtypes. To this aim, following the approach of [5], we introduce a notion of *normal form* of types such that:

- reduction to normal form preserves isomorphism;
- the normal form of a type is unique.

Normal forms are reached by applying as far as possible the following set of isomorphism preserving transformations, that are all defined by suitable η-expansions of the identity:

- the elimination of redundant intersections and unions, representing types that are intuitively (and provably) included, like the case of $(\sigma \to \tau) \wedge (\sigma \vee \rho \to \tau)$, that can clearly be reduced to $\sigma \vee \rho \to \tau$ or $(\sigma \to \rho \vee \tau) \wedge (\sigma \to \tau)$, that can be reduced to $\sigma \to \tau$ (*erasure*);
- the distribution of arrows over intersections and unions by elimination of intersection to the right of arrows and union to the left of arrows using the isomorphisms ($\to \wedge$**comm**) and ($\to \vee$**comm**) from left to right and the isomorphisms (**Dist** $\wedge\vee$**)** (*splitting*);
- the reduction of the so obtained types to unions of intersections of arrows, using and (**Dist** $\vee\wedge$**)** and (**Dist** $\wedge\vee$**)** (*distribution*).

For example the type $((\varphi_1 \wedge \varphi_2 \to \varphi_2 \vee \varphi_3) \wedge (\varphi_2 \to \varphi_5)) \vee ((\varphi_2 \wedge \varphi_3 \to \varphi_5) \wedge (\varphi_4 \to \varphi_3 \vee \varphi_5))$ is in normal form. Another example is $((\varphi_1 \wedge \varphi_2 \to \varphi_2) \wedge (\varphi_2 \to \varphi_5)) \vee ((\varphi_2 \to \varphi_5) \wedge (\varphi_4 \to \varphi_3 \vee \varphi_5) \to \varphi_6)$.

This normalisation process, although rather intuitive, needs some care. Because the transformations used to reduce a type to its normal form must be isomorphism preserving, each transformation has to be made in a type context $\mathcal{C}[\,]$ only if there are η-expansions of the identity proving the isomorphisms. For instance in the case of splitting we have that $\mathcal{C}[\sigma \to \tau \wedge \rho]$ reduces to $\mathcal{C}[(\sigma \to \tau) \wedge (\sigma \to \rho)]$ only if there is an η-expansions of the identity **Id** such that

$$\mathbf{Id} : \mathcal{C}[\sigma \to \tau \wedge \rho] \to \mathcal{C}[(\sigma \to \tau) \wedge (\sigma \to \rho)] \text{ and}$$
$$\mathbf{Id} : \mathcal{C}[\sigma \vee \tau \to \rho] \to \mathcal{C}[(\sigma \to \rho) \wedge (\tau \to \rho)]$$

Analogous considerations must be made for the other transformations.
For instance the types $(\sigma \vee \tau \to \rho) \wedge (\sigma \to \rho)$ and $\sigma \vee \tau \to \rho$ are isomorphic, while the types $(\sigma \vee \tau \to \rho) \wedge \varphi$ and $(\sigma \vee \tau \to \rho) \wedge (\sigma \to \rho) \wedge \varphi$, although semantically equal, are not; in fact $\sigma \vee \tau \to \rho \approx (\sigma \vee \tau \to$

$\rho) \wedge (\sigma \to \rho)$ is proved by the η-expansion of the identity $\lambda xy.xy$, but no η-expansion of the identity can map an atomic type into itself.

Note that an algorithm to find the normal form of an arbitrary type can be given, so the notion of normal form is effective.

Using Lemma 3.4 we can prove that if the normal forms of two types σ and τ are similar then $\sigma \approx \tau$. We conjecture that this also gives a complete characterisation of isomorphism, i.e. that if $\sigma \approx \tau$ then the normal forms of σ and τ are similar.

References

[1] Steffen van Bakel, Mariangiola Dezani-Ciancaglini, Ugo de' Liguoro, and Yoko Motohama. The minimal relevant logic and the call-by-value λ-calculus. Technical Report TR-ARP-05-2000, The Australian National University, 2000.

[2] Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Ugo de'Liguoro. Intersection and union types: Syntax and semantics. *Information and Computation*, 119:202–230, 1995.

[3] Kim Bruce and Giuseppe Longo. Provable isomorphisms and domain equations in models of typed languages. In Robert Sedgewick, editor, *STOC'85*, pages 263 – 272. ACM Press, 1985.

[4] Mariangiola Dezani-Ciancaglini. Characterization of normal forms possessing an inverse in the $\lambda\beta\eta$-calculus. *Theoretical Computer Science*, 2(3):323–337, 1976.

[5] Mariangiola Dezani-Ciancaglini, Roberto Di Cosmo, Elio Giovannetti, and Makoto Tatsuta. On isomorphisms of intersection types. *ACM Transactions on Computational Logic*, 11(4):1–22, 2010.

[6] Roberto Di Cosmo. A short survey of isomorphisms of types. *Mathematical Structures in Computer Science*, 15:825–838, 2005.

[7] David MacQueen, Gordon Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71(1-2):95–130, 1986.

[8] Gordon D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[9] Richard Routley and Robert K. Meyer. The semantics of entailment III. *Journal of Philosophical Logic*, 1:192–208, 1972.