On Isomorphisms of Intersection Types

Mariangiola Dezani-Ciancaglini[1], Roberto Di Cosmo[2],
Elio Giovannetti[1], Makoto Tatsuta[3]

[1] Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy
[2] Université Paris Diderot, PPS, UMR 7126, case 7014, 2 place Jussieu, 75005 Paris, France
[3] National Institute of Informatics, 2-1-2 Hitotsubashi, 101-8430 Tokyo, Japan

Abstract. The study of type isomorphisms for different λ-calculi started over twenty years ago, and a very wide body of knowledge has been established, both in terms of results and in terms of techniques. A notable missing piece of the puzzle was the characterization of type isomorphisms in the presence of intersection types. While at first thought this may seem to be a simple exercise, it turns out that not only finding the right characterization is not simple, but that the very notion of isomorphism in intersection types is an unexpectedly original element in the previously known landscape, breaking most of the known properties of isomorphisms of the typed λ-calculus. In particular, types that are equal in the standard models of intersection types may be non-isomorphic.

1 Introduction

The notion of *type isomorphism* is a particularization of the general notion of isomorphism as defined, for example, in category theory. Two objects σ and τ are *isomorphic* iff there exist two morphisms $f\colon \sigma \to \tau$ and $g\colon \tau \to \sigma$ such that $f \circ g = id_\tau$ and $g \circ f = id_\sigma$:

$$\begin{tikzcd} \sigma \arrow[loop left, "id_\sigma"] \arrow[r, bend left, "f"] & \tau \arrow[l, bend left, "g"] \arrow[loop right, "id_\tau"] \end{tikzcd}$$

Analogously, two *types* σ and τ in some (abstract) programming language, like the typed λ-calculus, are *isomorphic* if the same diagram holds, with f and g functions of types $\sigma \to \tau$ and $\tau \to \sigma$ respectively.

In the early 1980s, some interest started to develop in the problem of finding *all* the domain equations (type isomorphisms) that must hold in *every* model of a given language, or *valid isomorphisms of types*, as they were called in [4].

There are essentially two families of techniques for addressing this question: it is possible to work *syntactically* to characterize those programs f that possess an inverse g making the above diagram commute, or one can work *semantically* trying to find some specific model that captures the isomorphisms valid in all models (see [8] for a recent survey).

Each approach has its own difficulty: finding the syntactic characterization of the invertible terms can be very hard, while the rest follows then rather straightforwardly; finding the right specific model and showing that the only isomorphisms holding in it are those holding in all models can be very hard too, even if the advent of game semantics has a bit blurred the distinction between these approaches, by building models which are quite syntactical in nature [10].

Table 1 Type isomorphisms in typed lambda calculi

(swap)	$\sigma \to (\tau \to \gamma) = \tau \to (\sigma \to \gamma)$	$\}\, Th^1$		
1.	$\sigma \times \tau = \tau \times \sigma$			
2.	$\sigma \times (\tau \times \gamma) = (\sigma \times \tau) \times \gamma$			
3.	$(\sigma \times \tau) \to \gamma = \sigma \to (\tau \to \gamma)$			
4.	$\sigma \to (\tau \times \gamma) = (\sigma \to \tau) \times (\sigma \to \gamma)$	$\Big\}\, Th^1_{\times T}$ (1–7)		
5.	$\sigma \times \mathbf{T} = \sigma$			
6.	$\sigma \to \mathbf{T} = \mathbf{T}$		$\Big\}\, Th^2_{\times T}$ (1–12)	
7.	$\mathbf{T} \to \sigma = \sigma$			$\Big\}\, -10, 11 = Th^{ML}$
8.	$\forall X.\forall Y.\sigma = \forall Y.\forall X.\sigma$			
9.	$\forall X.\sigma = \forall Y.\sigma[Y/X]$	$\Big\}\, + \mathbf{swap}\ = Th^2$ (8–10)		
10.	$\forall X.(\sigma \to \tau) = \sigma \to \forall X.\tau$			
11.	$\forall X.\sigma \times \tau = \forall X.\sigma \times \forall X.\tau$			
12.	$\forall X.\mathbf{T} = \mathbf{T}$			
split	$\forall X.\sigma \times \tau = \forall X.\forall Y.\sigma \times (\tau[Y/X])$			

N.B.: in equation 8, X must be free for Y in σ and $Y \notin FTV(\sigma)$; in equation 10, $X \notin FTV(\sigma)$.

In our work, we started along the first line (as we already know the shape of the invertible terms), so here we only recall the relevant literature for the syntactic approach.

Type isomorphisms and invertible terms

In [6], Dezani fully characterized the *invertible λ-terms* as the *finite hereditary permutators*, a class of terms which can be easily defined inductively, and which can be seen as a family of generalized η-expansions.

Definition 1 (Invertible term). *A λ-term M is* invertible *if there exists a term M^{-1} such that $M \circ M^{-1} = M^{-1} \circ M =_{\beta\eta} \mathbf{I}$ (where $\circ$ denotes, as usual, functional composition, and $\mathbf{I}$ is the identity $\lambda x.x$). Obviously, M^{-1} is called an* inverse *of M.*

Definition 2 (Finite Hereditary Permutator). *A* finite hereditary permutator (f.h.p.) *is a λ-term whose head normal form is of the shape $\lambda z x_1 \dots x_n \,.\, z Q_1 \dots Q_n$ ($n \geq 0$) and is such that, for a permutation π of $1 \dots n$, the λ-terms $\lambda x_{\pi(1)}.Q_1, \dots, \lambda x_{\pi(n)}.Q_n$ are finite hereditary permutators.*

Theorem 1. *[6] A λ-term is invertible iff it is a finite hereditary permutator.*

Observe that f.h.p.'s are closed terms: so, by the above theorem, invertible λ-terms are closed terms.The proof of Theorem **??** shows that every f.h.p. has a unique inverse modulo $\beta\eta$-conversion. We use P to range over β-normal forms of f.h.p.'s. Thus P^{-1} denotes the unique (modulo η-conversion) inverse of P.

While the result of [6] was obtained in the framework of the untyped λ-calculus, it turned out that this family of invertible terms *can be typed* in the simply typed λ-calculus, and this allowed Bruce and Longo [4] to prove by a straightforward induction on the structure of the f.h.p.'s that in the simply typed λ-calculus the only type isomorphisms w.r.t. $\beta\eta$-equality are those induced by the *swap* equation

$$\sigma \to (\tau \to \rho) = \tau \to (\sigma \to \rho).$$

Notice that the type isomorphisms which correspond to invertible terms (called *definable isomorphisms of types in [4]*) are *a priori* not the same as the *valid isomorphisms of types*: a definable isomorphism seems to be a stronger notion, demanding that not only a given isomorphism holds in all models, but that it also holds in all models *uniformly*. Nevertheless, in all the cases studied in the literature, it is easy to build a free model out of the calculus, and to prove that valid and definable isomorphisms coincide, so this distinction has gradually disappeared in time, and in this work we will use the following definition of type isomorphism.

Definition 3 (Type isomorphism). *Given a λ-calculus along with a type system, two types σ and τ (in the system's type language) are* isomorphic*, and we write $\sigma \approx \tau$, if in the calculus there exists an invertible term, i.e., by the above theorem, a f.h.p.* P*, such that* $\vdash \mathsf{P} : \sigma \to \tau$ *and* $\vdash \mathsf{P}^{-1} : \tau \to \sigma$ *hold in the system. Following a standard nomenclature, we say that the term* P *proves the isomorphism $\sigma \approx \tau$, and we write $\sigma \approx_{\mathsf{P}} \tau$. Of course, $\sigma \approx_{\mathsf{P}} \tau$ iff $\sigma \approx_{\mathsf{P}^{-1}} \tau$.*

An immediate observation is that

Theorem 2. *Isomorphism is an equivalence relation.*

Observe that transitivity holds because invertible terms are closed under functional composition by definition. So if the f.h.p. P_1 proves $\sigma \approx \tau$ and the f.h.p. P_2 proves $\tau \approx \rho$, then $\mathsf{P}_2 \circ \mathsf{P}_1$ is a f.h.p. that proves $\sigma \approx \rho$.

By extending Dezani's original technique to the invertible terms in typed calculi with additional constructors (like products and unit type) or with higher order (System F or Core-ML), it has been possible to pursue this line of research to the point of getting a full characterization of isomorphisms in a whole set of typed λ-calculi, from $\lambda^1\beta\eta$, which corresponds to $IPC(\Rightarrow)$, the intuitionistic positive calculus with implication, whose isomorphisms are described by Th^1 [12,4], to $\lambda^1\beta\eta\pi*$, which corresponds to Cartesian Closed Categories and $IPC(\mathbf{True}, \wedge, \Rightarrow)$, for which $Th^1_{\times T}$ is complete [3][4], to $\lambda^2\beta\eta$ (System F), which corresponds to $IPC(\forall, \Rightarrow)$, and whose isomorphisms are given by Th^2 [4], to $\lambda^2\beta\eta\pi*$ (System F with products and unit type), which corresponds to $IPC(\forall, \mathbf{True}, \wedge, \Rightarrow)$, whose isomorphisms are given by $Th^2_{\times T}$ [7]. A summary of the axioms in these theories is given in Table 1.

Hence, in this line of research, the standard approach has been to find all the type isomorphisms for a given language (λ-calculus) and a given notion of equality on terms (which almost always contains extensional rules like η, as otherwise no nontrivial invertible term exists [6]) as a consequence of an inductive characterization of the invertible terms. The general schema in all the known cases is the same: guess an equational theory for the isomorphisms, find the invertible terms (this is the hard part), then by induction on their structure show the completeness of the equational theory (the easy part).

One notable missing piece in the table summarizing the theory of isomorphisms of

[4] But this result had been proved earlier by Soloviev using model theoretic techniques [14].

types is the case of intersection types. At first sight, it should be an easy exercise to deal with it: we already know the form of the invertible terms, as they are again the f.h.p.'s, and it should just be a matter of guessing the right equational theory and proving it complete by induction.

But it turns out that with intersection types all the intuitions that one has formed in the other systems fail: the intersection type discipline can give many widely different typings for the same term, so that the simple proof technique originated in [4] does not apply, and we are in for some surprises.

In this paper, we explore the world of type isomorphisms with intersection types, establishing a series of results that are quite unexpected: on the one hand, we will see in Section 2 that in the presence of intersection types the theory of isomorphisms is no longer a congruence, so that there is no hope to capture these isomorphisms via an equational theory, and the theory does not even include equality in the standard models; yet, decidability can be easily established, though with no simple bound on its complexity. On the other hand, we will be able to provide a very precise characterization of isomorphisms, via a special notion of similarity for type normal forms.

2 Basic Properties of Isomorphisms with Intersection Types

In this section we establish the basic properties of intersection types that show their deep difference with respect to the other cases studied in the literature, before tackling, in the later sections, their precise characterization.

Isomorphisms of intersection types are not a congruence

In all the cases known in the literature, the isomorphism equivalence relation is a *congruence*, as the type constructors explored so far (arrow, cartesian product, universal quantification, sum) all preserve isomorphisms.

Intersection, by contrast, does not preserve isomorphism: from $\sigma \approx \sigma'$ and $\tau \approx \tau'$ it does not follow, in general, that $\sigma \cap \tau \approx \sigma' \cap \tau'$. The intuitive reason is that the existence of two separate (invertible) functions that respectively transform all values of type σ into values of type σ' and all those of type τ into values of type τ', does not ensure that there is a function mapping any value that is both of type σ and of type τ to a value that is both of type σ' and of type τ'.

For example, though the isomorphism $\alpha \to \beta \to \gamma \quad\approx\quad \beta \to \alpha \to \gamma$ is given by the f.h.p. $\lambda xyz\,.\,xzy$, the two types $\varphi \cap (\alpha \to \beta \to \gamma)$ and $\varphi \cap (\beta \to \alpha \to \gamma)$ are not isomorphic, since the term $\lambda yz\,.\,xzy$ cannot be typed (from the assumption $x\colon \varphi$) with an atomic type φ, which can only be transformed into itself by the identity.

Therefore we have the following result:

Theorem 3. *The theory of isomorphisms for intersection types* is not a congruence.

In particular, this theory *cannot be described* with a standard equational theory: a non-trivial equivalence relation has to be devised[5].

[5] Notice that even in the very tricky case of the sum types, isomorphism is a congruence [9].

Isomorphisms do not contain equality in the standard intersection models

Another quite unconventional fact is that [6]

Theorem 4. *Types* equality *in the standard models of intersection types does not entail type isomorphisms.*

Proof. Take for example the two isomorphic types $\alpha \to \gamma$ and $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma)$. They are semantically coincident, because the type $\alpha \cap \beta \to \gamma$ is greater than $\alpha \to \gamma$, and therefore its presence in the intersection is useless.

Now, if we just add to both a seemingly innocent intersection with an atomic type, we obtain the two types $(\alpha \to \gamma) \cap \varphi$ and $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma) \cap \varphi$, which also have identical meanings but are not isomorphic: if they were, the isomorphism would be given by the f.h.p. $\lambda xy.xy$ because, while the identity is trivially able to map any intersection to each of its components (i.e., $\vdash \lambda x.x\colon \sigma \cap \tau \to \sigma$, $\vdash \lambda x.x\colon \sigma \cap \tau \to \tau$), the mapping in the opposite direction, from $\alpha \to \gamma$ to $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma)$, requires an *$\eta$-expansion* of the identity, as can be seen from the following derivation, where $\Gamma = x\colon \alpha \to \gamma, y\colon \alpha \cap \beta$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma \vdash x\colon \alpha\to\gamma \qquad \dfrac{\Gamma\vdash y\colon \alpha\cap\beta}{\Gamma\vdash y\colon\alpha}\,(\cap\text{E})}{\Gamma\vdash xy\colon\gamma}\,(\to\text{E})}{x\colon \alpha\to\gamma\vdash \lambda y.xy\colon \alpha\cap\beta\to\gamma}\,(\to\text{I}) \qquad \dfrac{\dfrac{\dots}{x\colon\alpha\to\gamma,\, y\colon\alpha\vdash xy\colon\gamma}\,(\to\text{E})}{x\colon\alpha\to\gamma\vdash\lambda y.xy\colon\alpha\to\gamma}\,(\to\text{I})}{x\colon\alpha\to\gamma\vdash\lambda y.xy\colon(\alpha\cap\beta\to\gamma)\cap(\alpha\to\gamma)}\,(\cap\text{I})}{\vdash \lambda xy.xy\colon (\alpha\to\gamma)\to(\alpha\cap\beta\to\gamma)\cap(\alpha\to\gamma)}\,(\to\text{I})$$

An η-expansion of the identity, however, cannot map an atomic type to itself; in particular, the judgment $x\colon (\alpha \to \gamma) \cap \varphi \vdash \lambda y.xy\colon \varphi$ cannot be derived, hence the term $\lambda xy.xy$ cannot be assigned the type $(\alpha \to \gamma) \cap \varphi \to (\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma) \cap \varphi$.

We could establish an isomorphism relation including the pair of types $(\alpha \to \gamma) \cap \varphi$ and $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma) \cap \varphi$ only by assuming, as in some models, that all atomic types are arrow types.

One could simply see this fact as a proof that the universal model – traditionally hard to find – where all and only the valid isomorphisms hold is not a standard model; but it is quite unconventional that *equality* in the standard models is not included in the isomorphism relation, and this really comes from the strong intensionality of intersection types.

Decidability

Despite the weird nature of isomorphisms with intersection types, it is easy to establish the following decidability result.

Theorem 5. *Isomorphisms of intersection types are decidable.*

Proof. Given two types σ and τ, a f.h.p. of type $\sigma \to \tau$ may have a number of top-level abstractions at most equal to the number of top-level arrows, and also every subterm of the f.h.p. cannot have, at each nesting level, more abstractions than the corresponding

[6] The standard models of intersection types are the models in which the arrow is interpreted as function space constructor and the intersection as set theoretic intersection.

number of arrows nested at that level. The number of f.h.p.'s that are candidate to prove the isomorphism $\sigma \approx \tau$ is therefore finite, and each of them can be checked whether it can be assigned the type $\sigma \to \tau$ [13].

3 The Type System and the Reduction to Type Normal Form

In order to keep the theory sufficiently manageable, we restrict arrow types to the ones ending with an atomic type: $\sigma := \alpha \cap \alpha \cdots \cap \alpha$, with $\alpha := \sigma \to \cdots \to \sigma \to \varphi$, since it is well known that such restriction does not alter the set of typeable terms [15].

The formal syntax of types therefore is:

$$\begin{array}{ll} \sigma := \alpha \mid \sigma \cap \sigma & \text{types} \\ \alpha := \varphi \mid \sigma \to \alpha & \text{atomic and arrow types} \end{array}$$

where φ denotes an atomic type. We use σ, τ, ρ to range over types, α, β, γ to range over arrow types, and $\varphi, \chi, \psi, \vartheta, \xi$ to range over atomic types. We will occasionally use roman letters to denote atomic types in complex examples.

Also, we consider types modulo idempotence, commutativity and associativity of $\cap$, so we can write $\bigcap_{i \in I} \sigma_i$ with finite I. We write $\sigma \equiv \tau$ if σ coincides with τ modulo idempotence, commutativity and associativity of $\cap$.

The type assignment system is the standard simple system with intersection types for the ordinary λ-calculus [5].

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

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

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

Since we do not allow an arrow type to have an intersection on the right-hand side, we must modify the formal definition of isomorphism. To this purpose, the following notation is useful.

Notation If $\tau = \bigcap_{i \in I} \alpha_i$, then $\vdash \mathsf{P} : \sigma \mapsto \tau$ is short for $\vdash \mathsf{P} : \sigma \to \alpha_i$ for all $i \in I$.

Definition 3 is then replaced by

Definition 4 (Isomorphism for the intersection type system). *Two intersection types σ and τ are* isomorphic *($\sigma \approx \tau$) if there exists a f.h.p.* P *such that* $\vdash \mathsf{P} : \sigma \mapsto \tau$ *and* $\vdash \mathsf{P}^{-1} : \tau \mapsto \sigma$.

Adopting a technique similar to one used by [7], we introduce a notion of *type normal form* along with an isomorphism-preserving reduction, and then we give the syntactic characterization of isomorphism on normal types only. We use reduction to eliminate redundant (arrow) types in intersections, i.e., those that are intersected with types intuitively included in them. For example, $(\sigma \to \alpha) \cap (\sigma \cap \tau \to \alpha)$ reduces to $(\sigma \to \alpha)$. The reduction relation is expressed with the help of some preliminary definitions.

Definition 5 (Intersection Occurrence). An intersection occurrence *of α in τ is defined inductively as follows (always considering, as stated at the beginning, $\cap$ modulo commutativity and associativity):*

- *if $\tau = \alpha \cap \sigma$, then the showed occurrence of α is an intersection occurrence;*
- *if $\tau = \rho_1 \to \ldots \to \rho_n \to \tau' \to \beta$, and α is an intersection occurrence in τ', then α is an intersection occurrence in τ;*
- *if $\tau = \beta \cap \sigma$ and α is an intersection occurrence in β, then α is an intersection occurrence in τ.*

Definition 6 (Erasure). *If α is an intersection occurrence in τ, then the* erasure of α in τ *(notation $|\tau|_\alpha$) is defined by:*

- *$|\tau|_\alpha = \sigma$ if $\tau = \alpha \cap \sigma$;*
- *$|\tau|_\alpha = \sigma_1 \to \ldots \to \sigma_n \to |\tau'|_\alpha \to \beta$ if $\tau = \sigma_1 \to \ldots \to \sigma_n \to \tau' \to \beta$ and α is an intersection occurrence in τ';*
- *$|\tau|_\alpha = |\beta|_\alpha \cap \sigma$ if $\tau = \beta \cap \sigma$ and α is an intersection occurrence in β.*

Definition 7 (Finite Hereditary Identity). *A* finite hereditary identity (f.h.i.) *is an η-expansion of $\lambda x.x$. We use* Id *to range over f.h.i.'s (Note that f.h.i.'s are particular forms of f.h.p.'s).*

We are now able to state the reduction rule.

Definition 8 (Reduction). *The* reduction rule *is*

$$\tau \rightsquigarrow |\tau|_\alpha$$

if there are a type sub-expression α and two f.h.i.s Id, Id' *such that $\vdash \mathsf{Id} : |\tau|_\alpha \mapsto \tau$ and $\vdash \mathsf{Id}' : \tau \mapsto |\tau|_\alpha$.*

It is immediate to see that reduction is confluent and terminating, thus defining a *type normal form*. Also, a type and its normal form are isomorphic by definition since a f.h.i. is a f.h.p. and all f.h.i. are mutually inverse.

Observe that, as noted in Section 1, redundant arrow types cannot be erased if they occur in intersections with atomic types, which prevent η-expansions of the identity to provide the isomorphism between the original type and the simplified type: thus, while we have $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma) \rightsquigarrow \alpha \to \gamma$, the type $(\alpha \cap \beta \to \gamma) \cap (\alpha \to \gamma) \cap \varphi$ does not reduce to $(\alpha \to \gamma) \cap \varphi$. For any type σ, the type $\sigma \cap \varphi$ (with φ atomic) is in normal form, since the atom φ blocks any reduction.

On the other hand, the type $\sigma = ((\alpha \cap \beta \to \psi) \to \varphi) \cap ((\alpha \to \psi) \cap \chi \to \varphi)$ reduces to the type $\gamma = (\alpha \cap \beta \to \psi) \to \varphi$ through the f.h.i. $\lambda xy.x(\lambda v.yv)$ (which coincides with its own inverse). Note that, as pointed out in Section 1, the mapping from σ to γ only needs the simple identity (we have $\vdash \lambda x.x : \sigma \to \gamma$), but the opposite mapping requires an *η-expansion* of the identity, so as to have the typing $\vdash \lambda xy.x(\lambda v.yv) : \gamma \mapsto \sigma$.

We may now introduce the key notion of our work, i.e., a *similarity* between types, which we will prove to be the desired syntactic counterpart of the notion of isomorphism.

Definition 9 (Similarity). *The* similarity *between two sequences of types $\langle \sigma_1, \ldots, \sigma_m \rangle$ and $\langle \tau_1, \ldots, \tau_m \rangle$, 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* $\langle\sigma_1,\ldots,\sigma_i\cap\sigma_{i+1},\ldots,\sigma_m\rangle \sim \langle\tau_1,\ldots,\tau_i\cap\tau_{i+1},\ldots,\tau_m\rangle$*;*
3. *if* $\langle\sigma_i^{(1)},\ldots,\sigma_i^{(m)}\rangle \sim \langle\tau_i^{(1)},\ldots,\tau_i^{(m)}\rangle$ *for* $1\le i\le n$*, then* $\langle\sigma_1^{(1)}\to\ldots\to\sigma_n^{(1)}\to\alpha^{(1)},\ldots,\sigma_1^{(m)}\to\ldots\to\sigma_n^{(m)}\to\alpha^{(m)}\rangle \sim$ $\langle\tau_{\pi(1)}^{(1)}\to\ldots\to\tau_{\pi(n)}^{(1)}\to\alpha^{(1)},\ldots,\tau_{\pi(1)}^{(m)}\to\ldots\to\tau_{\pi(n)}^{(m)}\to\alpha^{(m)}\rangle$*,* *where* π *is a permutation of* $1,\ldots,n$*.*

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

The intuitive meaning is that, for two intersection 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. The notion of similarity exactly catches such meaning.

For example, the two types $(\varphi_1\to\varphi_2\to\varphi_3\to\chi)\cap(\psi_1\to\psi_2\to\psi_3\to\vartheta)$ and $(\varphi_3\to\varphi_2\to\varphi_1\to\chi)\cap(\psi_2\to\psi_3\to\psi_1\to\vartheta)$ are not *similar* and thus (as we will prove) not isomorphic, while the corresponding types with cartesian product instead of intersection are. The reason is that, owing to the semantics of intersection, the same f.h.p. must be able to map all the conjuncts of one intersection to the corresponding conjuncts in the other intersection. In the example, there is obviously not one f.h.p. that maps both $\varphi_1\to\varphi_2\to\varphi_3\to\chi$ to $\varphi_3\to\varphi_2\to\varphi_1\to\chi$ and at the same time $\psi_1\to\psi_2\to\psi_3\to\vartheta$ to $\psi_2\to\psi_3\to\psi_1\to\vartheta$.
On the other hand, the two types

$$(\rho_1\to\rho_2\to\rho_3\to\alpha)\cap(\sigma_1\to\sigma_2\to\sigma_3\to\beta),$$
$$(\rho_2\to\rho_3\to\rho_1\to\alpha)\cap(\sigma_2\to\sigma_3\to\sigma_1\to\beta)$$

are similar (and therefore isomorphic), since the permutation is the same in the two components of the intersection.

A type like $(\sigma_1\to\ldots\to\sigma_n\to\alpha)\cap\varphi$ may only be similar (and thus isomorphic) to itself, since the presence of the atom φ in the intersection blocks the possibility of any permutation other than the identity in the conjunct type subexpression $\sigma_1\to\ldots\to\sigma_n\to\alpha$.

A more complex example of similar types is the following:

$\alpha_1\cap\alpha_2\sim\beta_1\cap\beta_2$,
where (indicating atomic types by roman letters):
$\alpha_1=(e\to f)\to(a\cap b\to c\to d)\cap(g\to b\to c)\to s\to t$
$\alpha_2=(h\to k)\cap(p\to q)\to(u\to v\to w)\to q\cap r\to(a\cap b\to z)$
$\beta_1=(c\to a\cap b\to d)\cap(b\to g\to c)\to s\to(e\to f)\to t$
$\beta_2=(v\to u\to w)\to q\cap r\to(h\to k)\cap(p\to q)\to(a\cap b\to z)$.

Note that the introduction of type sequences in the definition of similarity is needed in order to keep the correspondence between types in intersections. Consider, for example, the following two types:

$$\rho_1=(\sigma_1\cap\alpha\to\sigma_2\to\beta)\cap(\tau_1\to\tau_2\to\gamma),$$
$$\rho_2=(\sigma_2\to\sigma_1\to\beta)\cap(\tau_2\to\alpha\cap\tau_1\to\gamma).$$

They are not isomorphic, and are also not similar since the sequences $\langle\sigma_1\cap\alpha,\tau_1\rangle$, $\langle\sigma_1,\alpha\cap\tau_1\rangle$ are not. If, however, the definitions were given directly through intersection,

owing to the associativity of $\cap$ the two sequences would be represented by the same intersection $\sigma_1 \cap \alpha \cap \tau_1$, and the two types ρ_1, ρ_2 would therefore be similar.
An equivalent, slightly more algorithmic, definition of *similarity* may be given through a notion of *permutation tree*.

Definition 10 (Permutation Tree).
- *The empty tree $\varnothing$ is a permutation tree.*
- *$\langle \pi, [\Pi_1, \ldots, \Pi_n] \rangle$ is a permutation tree if π is a permutation of $1, \ldots, n$ and $\Pi_1, \ldots, \Pi_n$ are permutation trees.*

An example of a permutation tree is the tree $\Pi_0 = \langle (2,3,1), [\langle (2,1), [\varnothing, \varnothing] \rangle, \varnothing, \varnothing] \rangle$.
A more complex example is the tree Π defined as follows:

$$\Pi = \langle (2,3,1), [\Pi_1, \varnothing, \Pi_3] \rangle$$
where
$$\Pi_1 = \Big\langle (3,1,4,2), \Big[\varnothing, \varnothing, \langle (2,1), [\varnothing, \varnothing] \rangle, \langle (1,3,2), [\varnothing, \varnothing, \varnothing] \rangle \Big] \Big\rangle$$
$$\Pi_3 = \Big\langle (1,2,3), \Big[\langle (2,1), [\varnothing, \varnothing] \rangle, \varnothing, \langle (3,2,1,4), [\varnothing, \varnothing, \varnothing, \varnothing] \rangle \Big] \Big\rangle$$

A permutation tree is nothing but an abstract representation of a f.h.p. One may easily build the concrete f.h.p. corresponding to a permutation tree, by creating as many fresh variables as is the cardinality of the permutation and by recursively creating subterms that respectively have those variables as head variables, in the order specified by the permutation.
In the following definition **trm** is the recursive mapping: it takes a permutation tree and the name z of a fresh variable, and creates a term with free head variable z, which is the β-reduct of the corresponding f.h.p. applied to z. The top-level mapping **fhp** merely abstracts the head variable so as to transform the term into a f.h.p. proper.

Definition 11 (F.h.p. corresponding to a permutation tree).
The f.h.p. corresponding to a permutation tree Π is:

$\mathbf{fhp}(\Pi) = \lambda z.\mathbf{trm}(\Pi, z)$, *with z fresh variable;*
$\mathbf{trm}(\varnothing, z) = z$;
$\mathbf{trm}(\langle \pi, [\Pi_1, \ldots, \Pi_n] \rangle, z) = \lambda x_1 \ldots x_n \,.\, z\, \mathbf{trm}(\Pi_1, x_{\pi(1)}) \ldots \mathbf{trm}(\Pi_n, x_{\pi(n)})$
with $x_1 \ldots x_n$ fresh variables.

Examples.
The f.h.p. corresponding to the permutation tree $\Pi_0 = \langle (2,3,1), [\langle (2,1), [\varnothing, \varnothing] \rangle, \varnothing, \varnothing] \rangle$ is the term $\lambda z x_1 x_2 x_3 . z(\lambda u_1 u_2 . x_2 u_2 u_1) x_3 x_1$.
The f.h.p. corresponding to the permutation tree $\Pi = \langle (2,3,1), [\Pi_1, \varnothing, \Pi_3] \rangle$ of the example above is the term $P = \lambda z x_1 x_2 x_3 . z P_1 P_2 P_3$, where

$$P_1 = \lambda u_1 u_2 u_3 u_4 \,.\, x_2 u_3 u_1 (\lambda v_1 v_2 \,.\, u_4 v_2 v_1)(\lambda w_1 w_2 w_3 \,.\, u_2 w_1 w_3 w_2)$$
$$P_2 = x_3$$
$$P_3 = \lambda y_1 y_2 y_3 \,.\, x_1 (\lambda s_1 s_2 \,.\, y_1 s_2 s_1) y_2 (\lambda t_1 t_2 t_3 t_4 \,.\, y_3 t_3 t_2 t_1 t_4)$$

A permutation tree represents a tree of nested permutations: if we *apply* it to a type having a homologous tree structure, i.e., if we (are able to) recursively perform on the type all the permutations at all levels, we obtain a new type which is clearly *similar* to the original one. We therefore give the following natural definition.

Definition 12 (Application of a permutation tree to a type).
Application of a permutation tree is a partial map from types to types:

- $\varnothing(\sigma) = \sigma$
- $\langle\pi, [\Pi_1, \dots, \Pi_n]\rangle(\sigma_1 \to \cdots \to \sigma_n \to \alpha) = \Pi_1(\sigma_{\pi(1)}) \to \cdots \to \Pi_n(\sigma_{\pi(n)}) \to \alpha$
- $\Pi(\sigma \cap \tau) = \Pi(\sigma) \cap \Pi(\tau)$
- $\Pi(\sigma) = \text{undefined}$ *otherwise.*

Taking again one of the examples above, if

$$\begin{aligned}
\alpha_1 &= (e\to f) \to (a \cap b\to c\to d) \cap (g\to b\to c) \to s \to t\\
\alpha_2 &= (h\to k) \cap (p\to q) \to (u\to v\to w) \to q \cap r \to (a \cap b\to z)\\
\Pi_0 &= \langle(2, 3, 1), [\langle(2, 1), [\varnothing, \varnothing]\rangle, \varnothing, \varnothing]\rangle
\end{aligned}$$

then we have $\Pi_0(\alpha_1 \cap \alpha_2) = \beta_1 \cap \beta_2$, where

$$\begin{aligned}
\beta_1 &= (c\to a \cap b\to d) \cap (b\to g\to c) \to s \to (e\to f) \to t\\
\beta_2 &= (v\to u\to w) \to q \cap r \to (h\to k) \cap (p\to q) \to (a \cap b\to z)
\end{aligned}$$

With the other example, if we have:

$\sigma = \gamma_1 \to \gamma_2 \to \xi_3 \to \xi$
where

$$\begin{aligned}
\gamma_1 &= (\varphi_{11}\to\varphi_{12}\to\chi_1) \to \chi_2 \to (\varphi_{31}\to\varphi_{32}\to\varphi_{33}\to\varphi_{34}\to\chi_3) \to \chi\\
\gamma_2 &= \vartheta_1 \to (\psi_{21}\to\psi_{22}\to\psi_{23}\to\vartheta_2) \to \vartheta_3 \to (\psi_{41}\to\psi_{42}\to\vartheta_4) \to \vartheta
\end{aligned}$$

then $\Pi(\sigma) = \tau$, where

$\tau = \gamma_2' \to \xi_3 \to \gamma_1' \to \xi$
where

$$\begin{aligned}
\gamma_2' &= \vartheta_3 \to \vartheta_1 \to (\psi_{42}\to\psi_{41}\to\vartheta_4) \to (\psi_{21}\to\psi_{23}\to\psi_{22}\to\vartheta_2)\to\vartheta\\
\gamma_1' &= (\varphi_{12}\to\varphi_{11}\to\chi_1) \to \chi_2 \to (\varphi_{33}\to\varphi_{32}\to\varphi_{31}\to\varphi_{34}\to\chi_3) \to \chi
\end{aligned}$$

Two types can then be defined as equivalent when one can be obtained from the other (modulo idempotence, commutativity and associativity, as usual) by applying a permutation tree.

Definition 13 (Type permutation-equivalence).
Two types σ and τ are permutation-equivalent*, notation $\sigma \doteqdot \tau$, if $\exists\, \Pi \,.\, \Pi(\sigma) \equiv \tau$.*

It is trivial to see that if $\Pi(\sigma) \equiv \tau$, then there also exists an *inverse* permutation tree Π^{-1} such that $\Pi^{-1}(\tau) \equiv \sigma$.

It is easy to prove that $\sigma \sim \tau$ if and only if $\sigma \doteqdot \tau$, so that the latter equivalence merely is an alternative definition of the previously defined similarity. We will therefore always use the first notation.
As an immediate consequence of Definition 12, we have the following lemma.

Lemma 1. *If $\Pi(\sigma) \equiv \tau$, with $\Pi = \langle\pi, [\Pi_1, \dots, \Pi_n]\rangle$, then there exists a set I of indices such that σ and τ have the forms:*

$$\sigma \equiv \textstyle\bigcap_{i\in I}(\sigma_1^i \to \ldots \to \sigma_n^i \to \alpha^i), \quad \tau \equiv \textstyle\bigcap_{i\in I}(\tau_1{}^i \to \ldots \to \tau_n{}^i \to \alpha^i)$$

and for all $i \in I$, for $k = 1, \dots, n$, one has $\Pi_k(\sigma^i_{\pi(k)}) \equiv \tau^i_k$, therefore $\sigma^i_{\pi(k)} \sim \tau^i_k$.

Note that the above definitions of *similarity* are *not* equivalent to stating that, in the inductive case:

$$\bigcap_{i\in I}(\sigma_1^i \to \ldots \to \sigma_n^i \to \alpha^i) \ \sim\ \bigcap_{i\in I}(\tau_1{}^i \to \ldots \to \tau_n{}^i \to \alpha^i)$$

if there exists a permutation π such that

$$\forall i \in I \,.\, \tau_k^i \sim\ \sigma^i_{\pi(k)} \quad \text{and} \quad \bigcap_{i\in I}\tau_k^i \sim \bigcap_{i\in I}\sigma^i_{\pi(k)} \quad \text{for } k = 1,\ldots,n.$$

A counterexample is given by the following pair of types:

$$\sigma = (\beta_1 \to \alpha_1) \cap (\beta_2 \to \alpha_2) \cap (\beta_3 \to \alpha_3)$$
$$\tau = (\gamma_1 \to \alpha_1) \cap (\gamma_2 \to \alpha_2) \cap (\gamma_3 \to \alpha_3)$$

where

$$\beta_1 = \varphi \to \chi \to \psi \to \vartheta = \gamma_2$$
$$\beta_2 = \varphi \to \psi \to \chi \to \vartheta = \gamma_3$$
$$\beta_3 = \chi \to \varphi \to \psi \to \vartheta = \gamma_1$$

We have $\Pi_1(\beta_1) \equiv \gamma_1,\ \ \Pi_2(\beta_2) \equiv \gamma_2,\ \ \Pi_3(\beta_3) \equiv \gamma_3$, with

$$\Pi_1 = \langle(2,1,3),[\varnothing,\varnothing,\varnothing]\rangle, \quad \Pi_2 = \langle(1,3,2),[\varnothing,\varnothing,\varnothing]\rangle,$$
$$\Pi_3 = \langle(3,1,2),[\varnothing,\varnothing,\varnothing]\rangle,$$

and therefore $\beta_1 \sim \gamma_1$, $\beta_2 \sim \gamma_2$, $\beta_3 \sim \gamma_3$; also, $\beta_1 \cap \beta_2 \cap \beta_3 \ \sim\ \gamma_1 \cap \gamma_2 \cap \gamma_3$ since trivially $\beta_1 \cap \beta_2 \cap \beta_3 \equiv \gamma_1 \cap \gamma_2 \cap \gamma_3$. This, however, does not allow us to conclude that $\sigma \ \sim\ \tau$, since there exists no permutation tree Π such that $\Pi(\sigma) = \tau$ (because $\Pi(\sigma) = \Pi(\sigma_1) \cap \Pi(\sigma_2) \cap \Pi(\sigma_3)$ should hold), or, equivalently, since – following the first definition of similarity – the two sequences $\langle\beta_1,\beta_2,\beta_3\rangle$, $\langle\gamma_1,\gamma_2,\gamma_3\rangle$ $(= \langle\beta_3,\beta_1,\beta_2\rangle)$ are not similar. Accordingly, the two types σ and τ are not similar ($\sigma \not\sim \tau$), and thus, as will be proved by Theorem 8, not isomorphic ($\sigma \not\approx \tau$).

4 Standard Properties of the Type System

Our system, being a trivial restriction of the simple intersection type system, obviously has the well-known standard properties of the unrestricted system [1]. In particular, the Lemma 2, a generation lemma and the subject reduction property hold, and the proofs are standard.

Lemma 2. *If $\Gamma \vdash \lambda x.M : \sigma$, then σ cannot be an atomic type.*

Lemma 3 (Generation Lemma).
1. *If $x : \bigcap_{i\in I}\alpha_i \vdash x : \bigcap_{j\in J}\beta_j$, then $\{\beta_j \mid j \in J\} \subseteq \{\alpha_i \mid i \in I\}$.*
2. *If $\Gamma \vdash \lambda x.M : \bigcap_{i\in I}(\sigma_i \to \alpha_i)$, then for all $i \in I$: $\Gamma, x : \sigma_i \vdash M : \alpha_i$.*
3. *If $\Gamma \vdash MN : \alpha$, then there exists a type σ such that $\Gamma \vdash M : \sigma \to \alpha$ and $\Gamma \vdash N : \sigma$.*

Proof. The proof is by induction on derivations.

Theorem 6 (Subject Reduction). *If $\Gamma \vdash M : \sigma$ and $M \longrightarrow_\beta N$, then $\Gamma \vdash N : \sigma$.*

Proof. Standard.

The Lemmata 4, 5 and 6 state some useful properties of η-expansions of the identity and of permutators. In particular, Lemma 4 expresses a necessary condition for a f.h.i. to be typeable with an arrow type, and gives the forms of the types of its subterms. Lemma 5.1 says that a f.h.i. is able to map an intersection $\alpha \cap \beta$ to one of its components, for example α, only if it is able to map such component α to itself (which is not always the case, since the number of top-level arrows in α cannot be less than the number of

Table 2 Proof of Lemma 6

$x:\bigcap_{i\in I}\alpha_i \vdash \mathsf{P}x:\bigcap_{j\in J}\beta_j$	$\Longrightarrow$	$x:\bigcap_{i\in I}\alpha_i \vdash \lambda z_1\dots z_n.x(\mathsf{P}_1 z_{\pi(1)})\dots(\mathsf{P}_n z_{\pi(n)}):\bigcap_{j\in J}\beta_j$ by Theorem 6
	$\Longrightarrow$	$\forall j\in J.\ x:\bigcap_{i\in I}\alpha_i \vdash \lambda z_1\dots z_n.x(\mathsf{P}_1 z_{\pi(1)})\dots(\mathsf{P}_n z_{\pi(n)}):\beta_j$ by rule $(\cap E)$
	$\Longrightarrow$	$\forall j\in J.\ \beta_j=\sigma_1^{(j)}\to\dots\to\sigma_n^{(j)}\to\gamma^{(j)}$ for some $\sigma_1^{(j)},\dots,\sigma_n^{(j)},\gamma^{(j)}$ by Lemma 2
	$\Longrightarrow$	$\forall j\in J.\ \mathit{\Gamma}\vdash x(\mathsf{P}_1 z_{\pi(1)})\dots(\mathsf{P}_n z_{\pi(n)}):\gamma^{(j)}$ where $\mathit{\Gamma}=x:\bigcap_{i\in I}\alpha_i, z_1:\sigma_1^{(j)},\dots z_n:\sigma_n^{(j)}$ by Lemma 3(2)
	$\Longrightarrow$	$\forall j\in J.\ x:\bigcap_{i\in I}\alpha_i\vdash x:\tau_1^{(j)}\to\dots\to\tau_n^{(j)}\to\gamma^{(j)}$ & $z_{\pi(1)}:\sigma_{\pi(1)}^{(j)}\vdash \mathsf{P}_1 z_{\pi(1)}:\tau_1^{(j)}$ & … & $z_{\pi(n)}:\sigma_{\pi(n)}^{(j)}\vdash \mathsf{P}_n z_{\pi(n)}:\tau_n^{(j)}$ for some $\tau_1^{(j)},\dots,\tau_n^{(j)}$ by Lemma 3(3)
	$\Longrightarrow$	$\forall j\in J.\ \exists i_j\in I.\ \alpha_{i_j}=\tau_1^{(j)}\to\dots\to\tau_n^{(j)}\to\gamma^{(j)}$ by Lemma 3(1)
	$\Longrightarrow$	$\forall j\in J.\ \exists i_j\in I.\ x:\alpha_{i_j}\vdash\mathsf{P}x:\beta_j$ by rules $(\to E)$ and $(\to I)$.

top-level abstractions of the f.h.i.). Lemma 5.2 states the rather obvious fact that if a f.h.i. is able to map both the type σ to itself and the type τ to itself, then it also maps their intersection to itself.

Finally, Lemma 6 states that a f.h.p. P maps an intersection $\bigcap_{i\in I}\alpha_i$ to another intersection $\bigcap_{j\in J}\beta_j$, i.e., $\vdash \mathsf{P}\colon \bigcap_{i\in I}\alpha_i \mapsto \bigcap_{j\in J}\beta_j$, if and only if every component β_j in the target intersection is obtained by P from some component α_i in the source intersection.

In such lemmata and in the following we write judgments of the form $x\colon\sigma\vdash\mathsf{P}x\colon\tau$ (where P may also be Id) instead of $\vdash\mathsf{P}\colon\sigma\mapsto\tau$, in order to simplify the proofs. The two kinds of judgments are equivalent not because of a property of subject expansion, which does not hold in general, but because a f.h.p. P is an abstraction, and therefore $\vdash\mathsf{P}\colon\sigma\mapsto\tau$ if and only if $x\colon\sigma\vdash\mathsf{P}x\colon\tau$, as can be easily seen: if $x\colon\sigma\vdash\mathsf{P}x\colon\tau$, with $\mathsf{P}=\lambda x.M$, then by β-reduction and subject reduction one has $x\colon\sigma\vdash M\colon\tau$, whence, by ($\cap$E) and ($\to$I), $\vdash\mathsf{P}\colon\sigma\mapsto\tau$ (rule ($\cap$E) is needed since ($\to$I) can only build arrow types). More generally, this is a consequence of the property of subject expansion for intersection types in the λI-calculus. The opposite implication, from $\vdash\mathsf{P}\colon\sigma\mapsto\tau$ to $x\colon\sigma\vdash\mathsf{P}x\colon\tau$, trivially follows by ($\to$E) and ($\cap$I).

Lemma 4. *1. If $n\neq m$ or $\varphi\neq\varphi'$, then there is no* Id *such that*

$$\vdash\mathsf{Id}:(\sigma_1\to\dots\to\sigma_n\to\varphi)\to\tau_1\to\dots\to\tau_m\to\varphi'.$$

2. If $\vdash\mathsf{Id}:(\sigma_1\to\dots\to\sigma_n\to\varphi)\to\tau_1\to\dots\to\tau_n\to\varphi$ and $\sigma_m\neq\tau_m$ and $\sigma_q=\tau_q$ for $m+1\leq q\leq n$, then $\mathsf{Id}\ {}_{\beta}\!\longleftarrow\ \lambda y z_1\dots z_p.y(\mathsf{Id}_1 z_1)\dots(\mathsf{Id}_p z_p)$ for some p and some $\mathsf{Id}_1,\dots,\mathsf{Id}_p$ such that and $m\leq p\leq n$ and $\vdash\mathsf{Id}_l:\tau_l\mapsto\sigma_l$ for $1\leq l\leq p$.

Proof. Easy, using the Generation Lemma.

Lemma 5. *1. If $x:\sigma\cap\alpha\vdash\mathsf{Id}\,x:\alpha$, then $x:\alpha\vdash\mathsf{Id}\,x:\alpha$.*

2. If $x : \sigma \vdash \mathsf{Id}\, x : \sigma$ and $x : \tau \vdash \mathsf{Id}\, x : \tau$, then $x : \sigma \cap \tau \vdash \mathsf{Id}\, x : \sigma \cap \tau$.

Proof. Easy.

Lemma 6. $x : \bigcap_{i \in I} \alpha_i \vdash \mathsf{P}x : \bigcap_{j \in J} \beta_j$ *iff* $\forall j \in J.\ \exists i_j \in I.\ x : \alpha_{i_j} \vdash \mathsf{P}x : \beta_j$.

Proof. The right-to-left direction easily follows by application of the $(\cap E)$ rule and then of the $(\cap I)$ rule. For the left-to-right direction the proof is given in Table **??**, where $\mathsf{P}\ {}_{\beta}\!\longleftarrow \lambda y z_1 \ldots z_n . y(\mathsf{P}_1 z_{\pi(1)}) \ldots (\mathsf{P}_n z_{\pi(n)})$.

5 Isomorphism Characterisation

Having established an isomorphism-preserving reduction in Section 3, we can now restrict ourselves to normal types, for which we show that the similarity relation is a (sound and complete) characterization of isomorphism.

If we only consider normal types, we can strengthen the Lemma 6 by Lemma 8, which states that if a f.h.p. P has the type $\bigcap_{i \in I} \alpha_i \mapsto \bigcap_{j \in J} \beta_j$, then not only $\forall j \in J\, .\, \exists i_j \in I .\ \vdash \mathsf{P} : \alpha_{i_j} \to \beta_j$, but its inverse P^{-1} precisely maps each component β_j of the target intersection to its corresponding α_{i_j} in the source intersection. This is the key lemma that allows us to prove the main theorem, which states the coincidence between the two relations $\sim$ and $\approx$ for normal types.

Lemma 7 is instrumental to the proof of Lemma 8, and expresses the fact that in an intersection in normal form there are no redundant components, i.e., there cannot exist an η-expansion of the identity that "adds" one of the conjunct types starting from the others.

Lemma 7. *If $\tau \cap \alpha$ is normal, then there is no Id such that $x : \tau \vdash \mathsf{Id}\, x : \tau \cap \alpha$.*

Proof. Let $\tau = \bigcap_{i \in I} \alpha_i$. Towards a contradiction assume $x : \tau \vdash \mathsf{Id}\, x : \tau \cap \alpha$. Since $x : \tau \cap \alpha \vdash x : \tau$ we get $\tau \cap \alpha \rightsquigarrow \tau$.

Lemma 8. *If $\bigcap_{j \in J} \beta_j$ is a normal type, and $x : \bigcap_{i \in I} \alpha_i \vdash \mathsf{P}x : \bigcap_{j \in J} \beta_j$, and $x : \bigcap_{j \in J} \beta_j \vdash \mathsf{P}^{-1}x : \bigcap_{i \in I} \alpha_i$, and $x : \alpha_{i_0} \vdash \mathsf{P}x : \beta_{j_0}$, then $x : \beta_{j_0} \vdash \mathsf{P}^{-1}x : \alpha_{i_0}$.*

Proof. By Lemma 6 there is $j_1 \in J$ such that $x : \beta_{j_1} \vdash \mathsf{P}^{-1}x : \alpha_{i_0}$. We assume $j_0 \neq j_1$ towards a contradiction. From $x : \beta_{j_1} \vdash \mathsf{P}^{-1}x : \alpha_{i_0}$ and $x : \alpha_{i_0} \vdash \mathsf{P}x : \beta_{j_0}$ we get $x : \beta_{j_1} \vdash \mathsf{P}(\mathsf{P}^{-1}x) : \beta_{j_0}$, which implies $x : \bigcap_{j \in J, j \neq j_0} \beta_j \vdash (\mathsf{P} \circ \mathsf{P}^{-1})x : \bigcap_{j \in J} \beta_j$ by Lemma 5. This is, by Lemma 7, impossible, since $\mathsf{P} \circ \mathsf{P}^{-1}$ is β-reducible to a f.h.i.

Theorem 7 (Soundness of $\sim$). *If σ and τ are arbitrary types, then $\sigma \sim \tau$ implies $\sigma \approx \tau$.*

Proof. By induction on the definition of $\sim$ we show that $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$ implies that there is a f.h.p. P such that $\vdash \mathsf{P} : \sigma_j \mapsto \tau_j$ for $1 \leq j \leq m$.
The only interesting case is

$$\begin{array}{l} \langle \sigma_1, \ldots, \sigma_m \rangle = \langle \sigma_1^{(1)} \to \ldots \to \sigma_n^{(1)} \to \alpha^{(1)}, \ldots, \sigma_1^{(m)} \to \ldots \to \sigma_n^{(m)} \to \alpha^{(m)} \rangle \\ \langle \tau_1, \ldots, \tau_m \rangle = \langle \tau_{\pi(1)}^{(1)} \to \ldots \to \tau_{\pi(n)}^{(1)} \to \alpha^{(1)}, \ldots, \tau_{\pi(1)}^{(m)} \to \ldots \to \tau_{\pi(n)}^{(m)} \to \alpha^{(m)} \rangle, \end{array}$$

since $\langle \sigma_i^{(1)}, \ldots, \sigma_i^{(m)} \rangle \sim \langle \tau_i^{(1)}, \ldots, \tau_i^{(m)} \rangle$ for $1 \leq i \leq n$.
By induction, there is a P_i such that $\vdash \mathsf{P}_i : \sigma_i^{(j)} \mapsto \tau_i^{(j)}$ for $1 \leq j \leq m$. We can then choose P as the β-normal form of $\lambda y z_1 \ldots z_n . y(\mathsf{P}_1 z_{\pi^{-1}(1)}) \ldots (\mathsf{P}_n z_{\pi^{-1}(n)})$.

The opposite implication does not hold: two isomorphic types are not necessarily similar. For example, the type $\sigma = ((\alpha \cap \beta \rightarrow \psi) \rightarrow \varphi) \cap ((\alpha \rightarrow \psi) \cap \chi \rightarrow \varphi)$ and its normal form $\gamma = (\alpha \cap \beta \rightarrow \psi) \rightarrow \varphi$, already considered in Section 3, are isomorphic but not similar, simply because they are intersection types of different arities: γ consists of only one arrow type, while σ is an intersection of two arrow types, though one of them is redundant. On the other hand, the double implication holds for normal types.

Theorem 8 (Main Theorem). *If σ and τ are normal types, then $\sigma \approx \tau$ iff $\sigma \sim \tau$.*

Proof. We have to prove that $\sigma \approx \tau \implies \sigma \sim \tau$ (the opposite implication is established by Theorem 7).

We show by structural induction on P that if $\vdash \mathsf{P} \colon \sigma_j \mapsto \tau_j$ and $\vdash \mathsf{P}^{-1} \colon \tau_j \mapsto \sigma_j$ for $1 \leq j \leq m$, then $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$.
Let $\sigma_j = \bigcap_{1 \leq i \leq n_j} \alpha_i^{(j)}$ and $\tau_j = \bigcap_{1 \leq i \leq p_j} \beta_i^{(j)}$.
By Lemma 8 we get $n_j = p_j$ and $\vdash \mathsf{P} : \alpha_i^{(j)} \rightarrow \beta_i^{(j)}$ and $\vdash \mathsf{P}^{-1} : \beta_i^{(j)} \rightarrow \alpha_i^{(j)}$. Let $\mathsf{P} \;{}_{\beta}\!\!\longleftarrow \lambda y z_1 \ldots z_n . y(\mathsf{P}_1 z_{\pi(1)}) \ldots (\mathsf{P}_n z_{\pi(n)})$. As in the proof of Lemma 6, we get

$$\alpha_i^{(j)} = \tau_1^{(i,j)} \rightarrow \ldots \rightarrow \tau_n^{(i,j)} \rightarrow \gamma^{(i,j)} \text{ and } \beta_i^{(j)} = \sigma_1^{(i,j)} \rightarrow \ldots \rightarrow \sigma_n^{(i,j)} \rightarrow \gamma^{(i,j)}$$

and $\vdash \mathsf{P}_l : \sigma_{\pi(l)}^{(i,j)} \mapsto \tau_l^{(i,j)}$ and $\vdash \mathsf{P}_l^{-1} : \tau_l^{(i,j)} \mapsto \sigma_{\pi(l)}^{(i,j)}$ for $1 \leq l \leq n$. By induction we have

$$\begin{gathered}\langle \sigma_{\pi(l)}^{(1,1)}, \ldots, \sigma_{\pi(l)}^{(n_1,1)}, \ldots, \sigma_{\pi(l)}^{(1,m)}, \ldots, \sigma_{\pi(l)}^{(n_m,m)} \rangle \\ \sim \\ \langle \tau_l^{(1,1)}, \ldots, \tau_l^{(n_1,1)}, \ldots, \tau_l^{(1,m)}, \ldots, \tau_l^{(n_m,m)} \rangle\end{gathered}$$

for $1 \leq l \leq n$, which implies

$$\langle \alpha_1^{(1)}, \ldots, \alpha_{n_1}^{(1)}, \ldots, \alpha_1^{(m)}, \ldots, \alpha_{n_m}^{(m)} \rangle \sim \langle \beta_1^{(1)}, \ldots, \beta_{n_1}^{(1)}, \ldots, \beta_1^{(m)}, \ldots, \beta_{n_m}^{(m)} \rangle$$

and then $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$.

Of course, the characterization of isomorphisms immediately extends, via normalization, to all types of our system, as stated by the following corollary of the main theorem.

Theorem 9. *For any two types σ and τ, $\sigma \approx \tau \iff \sigma{\downarrow} \sim \tau{\downarrow}$, where $\sigma{\downarrow}$ and $\tau{\downarrow}$ are the normal forms respectively of σ and τ.*

Proof. Since a type is isomorphic to its normal form we have that:

1. for the $\Rightarrow$-direction, if $\sigma \approx \tau$, then $\sigma{\downarrow} \approx \sigma \approx \tau \approx \tau{\downarrow}$, whence, by the Main Theorem in the $\Rightarrow$-direction, $\sigma{\downarrow} \sim \tau{\downarrow}$;
2. for the opposite direction, if $\sigma{\downarrow} \sim \tau{\downarrow}$, then by the Main Theorem in the $\Leftarrow$-direction we have $\sigma{\downarrow} \approx \tau{\downarrow}$, whence: $\sigma \approx \sigma{\downarrow} \approx \tau{\downarrow} \approx \tau$, i.e., $\sigma \approx \tau$.

A prototypal isomorphism checker, directly obtained by the permutation-tree definition of similarity, has been realized in Prolog, and a simple web interface for it is available at the address `http://lambda.di.unito.it/iso/index.html`.

6 How to Normalise Types

The application of the type reduction rule, as defined in Section 3, suffers from combinatorial explosion in the search for the erasable type subexpression α, thus possibly making the normalization impractical. However, the search space can be considerably reduced with a more accurate formulation of the algorithm.

As explained in Section 3, the reduction may only simplify an intersection by erasing a type that is greater – according to the standard semantics – than one of the other conjuncts. We can then formally introduce a preorder relation on types, whose axioms and rules correspond to the view of "$\rightarrow$" as a function space constructor and of "$\cap$" as set intersection:

$$\sigma \leq \sigma \qquad \sigma \leq \tau,\ \tau \leq \rho\ \Rightarrow \sigma \leq \rho$$

$$\sigma \cap \tau \leq \sigma \qquad \sigma \cap \tau \leq \tau$$

$$\sigma \leq \tau,\ \sigma \leq \rho \Rightarrow \sigma \leq \tau \cap \rho$$

$$\sigma \leq \tau,\ \alpha \leq \beta \Rightarrow \tau \rightarrow \alpha \leq \sigma \rightarrow \beta$$

Then, when reducing a type σ to normal form, the search for a redundant type within σ may be limited to an outermost search for a type α that is greater than a type β in an intersection, followed by the testing whether two f.h.i. Id, Id' with the appropriate type exists, i.e., such that $\vdash \mathsf{Id}\colon |\sigma|_\alpha \mapsto \sigma$ and $\vdash \mathsf{Id}'\colon \sigma \mapsto |\sigma|_\alpha$. This can be performed through a mapping $\mathcal{I}$ which, applied to two types σ and τ, builds the set of all f.h.i.'s Id such that $\vdash \mathsf{Id}\colon\sigma \mapsto \tau$.

$$\begin{array}{ll}
\mathcal{I}(\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \varphi, \tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \varphi') = \varnothing & \\
 & \text{if } n \neq m \text{ or } \varphi \neq \varphi' \\
\mathcal{I}(\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \varphi, \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \varphi) & = \{\mathsf{Id} \mid \\
 & \lambda y z_1 \ldots z_p.y(\mathsf{Id}_1 z_1) \ldots (\mathsf{Id}_p z_p) \longrightarrow_\beta \mathsf{Id} \\
 & \&\ \mathsf{Id}_l \in \mathcal{I}(\tau_l, \sigma_l) \text{ for } 1 \leq l \leq p\} \\
 & \text{if } \sigma_m \neq \tau_m \text{ and } \sigma_q = \tau_q \\
 & \text{for } m+1 \leq q \leq n \text{ and } m \leq p \leq n
\end{array}$$

$$\mathcal{I}(\textstyle\bigcap_{i \in I} \alpha_i, \bigcap_{j \in J} \beta_j) = \{\mathsf{Id} \mid \forall j \in J \exists i \in I.\ \mathsf{Id} \in \mathcal{I}(\alpha_i, \beta_j)\}$$

The correctness of the mapping $\mathcal{I}$ easily follows from Lemmas 4 and 6.

7 Conclusions and Future Work

In this paper we have investigated for the first time the type isomorphisms for intersection types, and we have provided, by means of a fine analysis of the invertible terms, a precise characterization of their structure, despite the unexpected fact that isomorphism with intersection types is not a congruence.

Even if the isomorphism relation is decidable, we have shown that it is weaker than type equality in the standard models of intersection types, where arrows are interpreted as sets of functions, and intersections as set intersections; such equality is a congruence, consisting of the equality theory given by the axioms of commutativity, associativity and swap (i.e, the first line and the axioms 1 and 2 of Table 1 with $\times$ replaced by $\cap$) and by the order relation induced by the preorder reported in Section 6. This means that the *universal model for type isomorphisms* is not a standard model of intersection types, while Cartesian Closed Categories build a universal model for the simply typed lambda calculus with surjective pairing and terminal object; the existence of such natural universal model for intersection types is an open question.

Finally, we recall that since types may in general be interpreted – owing to the well-known Curry-Howard correspondence – as propositions in some suitable logic, a characterization of type isomorphisms may immediately become a characterization of strong logical equivalences between propositions. In the case of intersection types, however, this is a problematic issue, since it is well known that intersection is an intensional operator, with no direct logical counterpart in the Curry-Howard sense. Recently, new kinds of logics have been proposed which give a logical meaning to the intersection operator [2], [11]. It might therefore be interesting to explore the role of intersection type isomorphisms in such contexts.

Acknowledgments. We would like to thank the anonymous referees for their detailed remarks and helpful comments.

References

1. H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940 (1984), 1983.
2. V. Bono, B. Venneri, and L. Bettini. A typed lambda calculus with intersection types. *Theoretical Computer Science*, 2008. To appear.
3. K. Bruce, R. Di Cosmo, and G. Longo. Provable isomorphisms of types. *Mathematical Structures in Computer Science*, 2(2):231–247, 1992.
4. K. Bruce and G. Longo. Provable isomorphisms and domain equations in models of typed languages. In R. Sedgewick, editor, *STOC'85*, pages 263 – 272. ACM, 1985.
5. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.
6. M. Dezani-Ciancaglini. Characterization of normal forms possessing an inverse in the $\lambda\beta\eta$ calculus. *Theoretical Computer Science*, 2:323–337, 1976.
7. R. Di Cosmo. Second order isomorphic types. A proof theoretic study on second order λ-calculus with surjective pairing and terminal object. *Information and Computation*, pages 176–201, 1995.
8. R. Di Cosmo. A short survey of isomorphisms of types. *Mathematical Structures in Computer Science*, 15:825–838, 2005.
9. M. Fiore, R. Di Cosmo, and V. Balat. Remarks on isomorphisms in typed lambda calculi with empty and sum types. *Annals of Pure and Applied Logic*, 141(1–2):35–50, 2006.
10. O. Laurent. Classical isomorphisms of types. *Mathematical Structures in Computer Science*, 15:969–1004, 2005.
11. L. Liquori and S. Ronchi Della Rocca. Intersection types a la Church. *Information and Computation*, 205(9):1371–1386, 2007.
12. C. F. Martin. Axiomatic bases for equational theories of natural numbers. *Notices of the American Mathematical Society*, 19(7):778, 1972.
13. S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:1–29, 1988.
14. S. Soloviev. A complete axiom system for isomorphism of types in closed categories. In A. Voronkov, editor, *LPAR'93*, volume 698 of *Lecture Notes in Computer Science*, pages 360–371. Springer-Verlag, 1993.
15. S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.