On Isomorphisms of Intersection Types

Mariangiola Dezani-Ciancaglini
Roberto Di Cosmo
Elio Giovannetti
and
Makoto Tatsuta

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, isomorphism is not a congruence and types that are equal in the standard models of intersection types may be non-isomorphic.

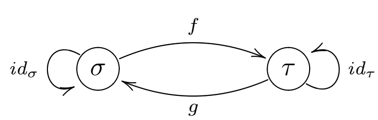
Categories and Subject Descriptors: F.4.1 [**Theory of Computation**]: Mathematical Logic—*Lambda calculus and related systems*; F.3.3 [**Theory of Computation**]: Studies of Program Constructs—*Type structure*; D.1.1 [**Software**]: Applicative (Functional) Programming

General Terms: Theory, Languages

Additional Key Words and Phrases: Type Isomorphism, Lambda calculus, Intersection Types

1. INTRODUCTION

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



Author addresses: Mariangiola Dezani-Ciancaglini, Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy, Roberto Di Cosmo, Université Paris Diderot, PPS, UMR 7126, case 7014, 2 place Jussieu, 75005 Paris, France, Elio Giovannetti, Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy, Makoto Tatsuta, National Institute of Informatics, 2-1-2 Hitotsubashi, 101-8430 Tokyo, Japan.

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 $\sigma \rightarrow \tau$ and $\tau \rightarrow \sigma$ 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 [Bruce and Longo 1985].

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 [Di Cosmo 2005] 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 [Laurent 2005].

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 [Dezani-Ciancaglini 1976], 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 1.1 *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, i.e. $N_1 \circ N_2 = \lambda x.N_1(N_2 x)$, and $\mathbf{I}$ is the identity $\lambda x.x$). Obviously, M^{-1} is called an *inverse* of M.

Definition 1.2 *Finite Hereditary Permutator.* A λ-term is a *finite hereditary permutator (f.h.p.)* when its β-normal form is $\lambda x y_1 \ldots y_n.x Q_1 \ldots Q_n$ ($n \geq 0$) and is such that, for a permutation π of $1 \ldots n$, the λ-terms $\lambda y_{\pi(1)}.Q_1, \ldots, \lambda y_{\pi(n)}.Q_n$ are finite hereditary permutators.

An example of an f.h.p. is $\lambda x y_1 y_2 y_3.x y_2 (\lambda z_1 z_2.y_3 z_2 z_1) y_1$, and an inverse of this λ-term is $\lambda x y_1 y_2 y_3.x y_3 y_1 (\lambda z_1 z_2.y_2 z_2 z_1)$.

THEOREM 1.3. *[Dezani-Ciancaglini 1976] 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 reduce to closed terms. The proof of Theorem 1.3 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.

[†]A model of a set of types is a domain equipped with an interpretation function mapping each type to a domain element.

While the result of [Dezani-Ciancaglini 1976] 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 [Bruce and Longo 1985] 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 [Bruce and Longo 1985]*) are *a priori* not the same as the *valid isomorphisms of types*: a de nable 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 de nable isomorphisms coincide, so this distinction has gradually disappeared in time, and in this work we will use the following de nition of type isomorphism.

Definition 1.4 *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, an 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, $\sigma \approx_{\mathsf{P}} \tau$ i $\sigma \approx_{\mathsf{P}^{-1}} \tau$.

An immediate observation is that

THEOREM 1.5. *Isomorphism is an equivalence relation.*

Observe that transitivity holds because invertible terms are closed under functional composition by de nition. 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 an 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 product and unit type) or with higher order types (System F), 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 [Martin 1972; Bruce and Longo 1985], to $\lambda^1\beta\eta\pi*$, which corresponds to Cartesian Closed Categories and $IPC(\mathbf{True}, \wedge, \Rightarrow)$, for which $Th^1{}_T$ is complete [Bruce et al. 1992][‡], to $\lambda^2\beta\eta$ (System F), which corresponds to $IPC(\forall, \Rightarrow)$, and whose isomorphisms are given by Th^2 [Bruce and Longo 1985], and 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 $Th^2{}_T$ [Di Cosmo 1995]. A summary of the axioms in these theories is given in Table I.

‡But this result had been proved earlier by Soloviev using model-theoretic techniques [Soloviev 1983; **?**].

Table I Type isomorphisms in typed lambda calculi

(swap) $\sigma \to (\tau \to \gamma) = \tau \to (\sigma \to \gamma)$	Th^1			
1. $\sigma \times \tau = \tau \times \sigma$	$Th^1{}_T$	$Th^2{}_T$	$- 10, 11 = Th^{ML}$	
2. $\sigma \times (\tau \times \gamma) = (\sigma \times \tau) \times \gamma$				
3. $(\sigma \times \tau) \to \gamma = \sigma \to (\tau \to \gamma)$				
4. $\sigma \to (\tau \times \gamma) = (\sigma \to \tau) \times (\sigma \to \gamma)$				
5. $\sigma \times \mathbf{T} = \sigma$				
6. $\sigma \to \mathbf{T} = \mathbf{T}$				
7. $\mathbf{T} \to \sigma = \sigma$				
8. $\forall X.\forall Y.\sigma = \forall Y.\forall X.\sigma$	+ **swap** $= Th^2$			
9. $\forall X.\sigma = \forall Y.\sigma[Y/X]$				
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 9, Y does not occur free in σ and the substitution must be capture avoiding; in equation 10, X does not occur free in σ.

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 [Dezani-Ciancaglini 1976]) as a consequence of an inductive characterization of the invertible terms. The general schema in all the known cases is the same: first guess an equational theory for the isomorphisms (this is the hard part), then by induction on the structure of the invertible terms show the completeness of the equational theory (the easy part).

One notable missing piece in the table summarizing the theory of isomorphisms of 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 [Bruce and Longo 1985] 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 in the following sections a very precise characterization of isomorphisms, via a special notion of similarity for type normal forms.

The present paper is an expanded version of [Dezani-Ciancaglini et al. 2008]. The main di erence concerns the syntax of intersection types, which in [Dezani-Ciancaglini et al. 2008] generated only arrow types ending with atomic types. This syntactic restriction considerably simpli ed the characterization of isomorphisms, since only one reduction rule was enough to get the normal forms of types, while here we need a further reduction rule whose applicability condition was quite di cult to devise.

2. BASIC PROPERTIES OF ISOMORPHISMS WITH INTERSECTION TYPES

In this section we recall the intersection type discipline, and establish the basic properties of intersection types that show their deep di erence w.r.t. the other cases studied in the literature, before tackling, in the later sections, their precise characterization.

2.1 Intersection types

The formal syntax of intersection types is:

$$\sigma \ := \ \varphi \quad | \ \sigma \to \sigma \ | \ \sigma \cap \sigma$$

where φ denotes an atomic type. We use σ, τ, ρ to range over types, μ, ν, λ to range over atomic and arrow 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. We shall use the convention that $\cap$ takes precedence over $\to$ and that $\to$ associates to the right.

Also, we consider types modulo idempotence, commutativity and associativity of $\cap$, so we can write $\bigcap_{i2I}\sigma_i$ with nite I. This is sound since clearly idempotence, commutativity and associativity of $\cap$ preserves type isomorphism, in fact:
$\vdash \lambda x.x{:}\sigma \to \sigma \cap \sigma$, $\vdash \lambda x.x{:}\sigma \cap \sigma \to \sigma$, $\vdash \lambda x.x{:}\sigma \cap \tau \to \tau \cap \sigma$,
$\vdash \lambda x.x{:}(\sigma \cap \tau) \cap \rho \to \sigma \cap (\tau \cap \rho)$ and $\vdash \lambda x.x{:}\sigma \cap (\tau \cap \rho) \to (\sigma \cap \tau) \cap \rho$.
We write $\sigma \equiv \tau$ if σ coincides with τ modulo idempotence, commutativity and associativity of $\cap$.

The type assignment system is the standard system with intersection types for the ordinary λ-calculus [Coppo and Dezani-Ciancaglini 1980].

$$(Ax) \qquad , x{:}\sigma \vdash x{:}\sigma$$

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

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

2.2 Isomorphisms of intersection types are not a congruence

In all the cases known in the literature, the isomorphism equivalence relation of De nition 1.4 is a *congruence*, as the type constructors explored so far (arrow, cartesian product, universal quanti cation, sum) all preserve isomorphisms.

Intersection, by contrast, does not preserve isomorphism: from $\sigma \approx \sigma^{0}$ and $\tau \approx \tau^{0}$ it does not follow, in general, that $\sigma \cap \tau \approx \sigma^{0} \cap \tau^{0}$. 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^{0} and all those of type τ into values of type τ^{0}, 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^{0} and of type τ^{0}. It is also worthwhile to notice that set theoretic isomorphisms are not preserved by intersections.

For example, though the isomorphism $\sigma \to \tau \to \rho \;\approx\; \tau \to \sigma \to \rho$ is given by the f.h.p. $\lambda xyz.xzy$, the two types $\varphi \cap (\sigma \to \tau \to \rho)$ and $\varphi \cap (\tau \to \sigma \to \rho)$ 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 2.1. *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[x].

2.3 Isomorphisms do not contain equality in the standard intersection models

Another quite unconventional fact is that

THEOREM 2.2. *Type* equality *in the standard models* [{] *of intersection types does not entail type isomorphisms.*

PROOF. Take for example the two isomorphic types

$$\sigma \to \rho \quad \text{and} \quad (\sigma \cap \tau \to \rho) \cap (\sigma \to \rho).$$

They are semantically coincident, because the type $\sigma \cap \tau \to \rho$ is greater than $\sigma \to \rho$, 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 $(\sigma \to \rho) \cap \varphi$ and $(\sigma \cap \tau \to \rho) \cap (\sigma \to \rho) \cap \varphi$, 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., $\vdash \lambda x.x\colon \sigma_1 \cap \sigma_2 \to \sigma_1$, $\vdash \lambda x.x\colon \sigma_1 \cap \sigma_2 \to \sigma_2$), the mapping in the opposite direction, from $\sigma \to \rho$ to $(\sigma \cap \tau \to \rho) \cap (\sigma \to \rho)$, requires an *$\eta$-expansion* of the identity, as can be seen from the following derivation, where $= x\colon \sigma \to \rho, y\colon \sigma \cap \tau$:

§Notice that even in the very tricky case of the sum types, isomorphism is a congruence [Fiore et al. 2006].

¶The standard models of intersection types map types to subsets of any domain that is a model of the untyped lambda calculus, with the condition that the arrow is interpreted as function space constructor and the intersection as set-theoretic intersection. I.e., the interpretation of $\sigma \;!\; \tau$ is the set of functions which map every element belonging to (the interpretation of) σ to an element belonging to (the interpretation of) τ.

$$
\dfrac{
  \dfrac{
    \dfrac{\vdash x\colon \sigma\to\rho \qquad \dfrac{\vdash y\colon \sigma\cap\tau}{\vdash y\colon \sigma}\,(\cap\mathsf{E})}{\vdash xy\colon \rho}\,(\to\mathsf{E})
  }{x\colon \sigma\to\rho \vdash \lambda y.xy\colon \sigma\cap\tau\to\rho}\,(\to\mathsf{I})
  \qquad
  \dfrac{
    \dfrac{\dots}{x\colon \sigma\to\rho,\, y\colon \sigma \vdash xy\colon \rho}\,(\to\mathsf{E})
  }{x\colon \sigma\to\rho \vdash \lambda y.xy\colon \sigma\to\rho}\,(\to\mathsf{I})
}{
  \dfrac{x\colon \sigma\to\rho \vdash \lambda y.xy\colon (\sigma\cap\tau\to\rho)\cap(\sigma\to\rho)}{\vdash \lambda xy.xy\colon (\sigma\to\rho)\to(\sigma\cap\tau\to\rho)\cap(\sigma\to\rho)}\,(\to\mathsf{I})
}\,(\cap\mathsf{I})
$$

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

We could establish an isomorphism relation including the pair of types $(\sigma\to\rho)\cap\varphi$ and $(\sigma\cap\tau\to\rho)\cap(\sigma\to\rho)\cap\varphi$ 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 nd { 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.

2.4 Decidability

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

THEOREM 2.3. *Isomorphisms of intersection types are decidable.*

PROOF. Given two types σ and τ, an 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 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 nite, and each of them can be checked whether it can be assigned the type $\sigma\to\tau$ [Ronchi Della Rocca 1988]. □

3. REDUCTION TO TYPE NORMAL FORM

Adopting a technique similar to one used by [Di Cosmo 1995], we introduce a notion of *type normal form* along with an isomorphism-preserving reduction, and then we give the syntactic characterization of isomorphisms on normal types only. We use reduction to:

- distribute arrows over intersections (*splitting*) and
- eliminate redundant (arrow) types in intersections (*erasure*), i.e., those types that are intersected with types intuitively included in them.

For example, $\sigma\to\tau\cap\rho$ reduces to $(\sigma\to\tau)\cap(\sigma\to\rho)$ by splitting, and $(\sigma\cap\rho\to\tau)\cap(\sigma\to\tau)$ reduces to $\sigma\to\tau$ by erasure.

The reduction relation is expressed with the help of some preliminary de nitions, in which, as stated at the beginning, we always consider $\cap$ modulo commutativity and associativity. The syntax of type contexts with one hole is as expected:

$$\mathcal{C}[\,] \;:=\; [\,] \mid \mathcal{C}[\,]\to\sigma \mid \sigma\to\mathcal{C}[\,] \mid \sigma\cap\mathcal{C}[\,].$$

3.1 Splitting

In order to decide when an occurrence of an arrow type $\sigma \to \tau \cap \rho$ inside a type context $\mathcal{C}[\ ]$ can be split we must check if $\mathcal{C}[\ ]$ has \enough arrows" to allow us to nd f.h.p.'s which map $\mathcal{C}[\sigma \to \tau \cap \rho]$ into $\mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)]$ and vice versa. To this aim the notion of *path* is handy. A path is a possibly empty list of positive naturals, which we denote by $\langle n_1, \ldots, n_m \rangle$ ($m \geq 0$). We use p to range over paths. The agreement of a type with a path holds when the type has the arrows required by the path (De nition 3.1). The path of a type context is de ned by requiring the agreement of some subtypes of the context with suitable subpaths (De nition 3.2).

Definition 3.1. The agreement of a type σ with a path p (notation $\sigma \propto \mathsf{p}$) is the smallest relation between types and paths such that:

- $\sigma \propto \langle\ \rangle$ for all σ;
- $\tau \propto \langle n_1, \ldots, n_m \rangle$ implies $\tau \to \rho \propto \langle 1, n_1, \ldots, n_m \rangle$;
- $\rho \propto \langle n_1, \ldots, n_m \rangle$ implies $\tau \to \rho \propto \langle n_1 + 1, \ldots, n_m \rangle$;
- $\tau \propto \langle n_1, \ldots, n_m \rangle$ and $\rho \propto \langle n_1, \ldots, n_m \rangle$ imply $\tau \cap \rho \propto \langle n_1, \ldots, n_m \rangle$.

For example the type $\sigma_1 \to (\sigma_2 \to \rho_1 \cap \rho_2) \cap (\sigma_3 \to \tau_1) \to \tau_2$ agrees with the path $\langle 2, 1 \rangle$, while the type $\sigma_1 \to (\sigma_2 \to \rho_1 \cap \rho_2) \cap (\sigma_3 \to \tau_1) \cap \varphi \to \tau_2$ does not agree with the path $\langle 2, 1 \rangle$, since φ does not agree with $\langle 1 \rangle$.

It is easy to verify that if a type agrees with a path, then it agrees with all its initial sub-paths, i.e. $\sigma \propto \langle n_1, \ldots, n_{m'}, \ldots, n_m \rangle$ implies $\sigma \propto \langle n_1, \ldots, n_{m'} \rangle$.

Definition 3.2. The path of a type context $\mathcal{C}[\ ]$ (notation $\mathsf{p}(\mathcal{C}[\ ])$) is de ned by induction on $\mathcal{C}[\ ]$:

- $\mathsf{p}(\mathcal{C}[\ ]) = \langle 1 \rangle$ if $\mathcal{C}[\ ] = [\ ]$;
- $\mathsf{p}(\mathcal{C}[\ ]) = \langle 1, n_1, \ldots, n_m \rangle$ if $\mathcal{C}[\ ] = \mathcal{C}'[\ ] \to \sigma$ and $\mathsf{p}(\mathcal{C}'[\ ]) = \langle n_1, \ldots, n_m \rangle$;
- $\mathsf{p}(\mathcal{C}[\ ]) = \langle n_1 + 1, \ldots, n_m \rangle$ if $\mathcal{C}[\ ] = \sigma \to \mathcal{C}'[\ ]$ and $\mathsf{p}(\mathcal{C}'[\ ]) = \langle n_1, \ldots, n_m \rangle$;
- $\mathsf{p}(\mathcal{C}[\ ]) = \mathsf{p}(\mathcal{C}'[\ ])$ if $\mathcal{C}[\ ] \equiv \mathcal{C}'[\ ] \cap \sigma$ and $\sigma \propto \mathsf{p}(\mathcal{C}'[\ ])$.

For example the path of the context $\sigma_1 \to [\ ] \cap (\sigma_2 \to \tau_1) \to \tau_2$ is $\langle 2, 1 \rangle$, while the path of the context $\sigma_1 \to [\ ] \cap (\sigma_2 \to \tau_1) \cap \varphi \to \tau_2$ is unde ned, since $\varphi \not\propto \langle 1 \rangle$.

One can easily show that $\mathsf{p}(\mathcal{C}[\ ])$ is de ned if and only if $\mathcal{C}[\sigma \to \tau] \propto \mathsf{p}(\mathcal{C}[\ ])$ for arbitrary σ and τ.

Definition 3.3 *Reduction by Splitting.* The *splitting reduction rule* is:

$$\mathcal{C}[\sigma \to \tau \cap \rho] \rightsquigarrow \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)]$$

if $\mathsf{p}(\mathcal{C}[\ ])$ is de ned.

For example $\sigma \to \tau \cap \rho \rightsquigarrow (\sigma \to \tau) \cap (\sigma \to \rho)$, while $\varphi \cap (\sigma \to \tau \cap \rho)$ cannot be reduced. It is easy to verify that $\lambda xy.xy$ shows the isomorphism $\sigma \to \tau \cap \rho \approx (\sigma \to \tau) \cap (\sigma \to \rho)$.

We will prove the soundness of this rule in Section 5, i.e., that if
$\mathcal{C}[\sigma \to \tau \cap \rho] \rightsquigarrow \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)]$, then there is P such that
$\vdash \mathsf{P} : \mathcal{C}[\sigma \to \tau \cap \rho] \to \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)]$ and
$\vdash \mathsf{P}^{-1} : \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)] \to \mathcal{C}[\sigma \to \tau \cap \rho]$.

3.2 Erasure

The condition for erasing an arrow type in an intersection uses particular forms of f.h.p.'s de ned as follows.

Definition 3.4 *Finite Hereditary Identity.* A *finite hereditary identity (f.h.i.)* is a β-normal form obtained from $\lambda x.x$ through a nite (possibly zero) number of η-expansions. We use Id to range over f.h.i.'s.

An example of f.h.i. is $\lambda xy.x(\lambda z.yz)$.

Definition 3.5 *Reduction by Erasure.* The *erasure reduction rule* is:

$$\mathcal{C}[\alpha \cap \sigma] \rightsquigarrow \mathcal{C}[\sigma]$$

if there are two f.h.i.s Id, Id^0 such that $\vdash$ Id$:\mathcal{C}[\alpha\cap\sigma] \to \mathcal{C}[\sigma]$ and $\vdash$ Id$^0:\mathcal{C}[\sigma] \to \mathcal{C}[\alpha\cap\sigma]$.

If we apply the erasure rule, then we only get isomorphic types by de nition, since an f.h.i. is an f.h.p. and every f.h.i. is its inverse.

We can use both splitting and erasure to reduce types, for example: $(\sigma \to \tau) \cap (\sigma \to \tau \cap \rho) \rightsquigarrow (\sigma \to \tau) \cap (\sigma \to \tau) \cap (\sigma \to \rho) \rightsquigarrow (\sigma \to \tau) \cap (\sigma \to \rho)$, and also $(\sigma \to \tau) \cap (\sigma \to \tau \cap \rho) \rightsquigarrow \sigma \to \tau \cap \rho \rightsquigarrow (\sigma \to \tau) \cap (\sigma \to \rho)$.

Note that $\vdash$ Id $: \mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$ does not imply that there is Id^0 such that $\vdash$ Id$^0:\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$, for example $\vdash \lambda x.x : \sigma \cap \alpha \to \sigma$. Also $\vdash$ Id$:\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ does not imply that there is Id^0 such that $\vdash$ Id$^0 : \mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$, for example $\vdash \lambda xy.xy : (\sigma \to \tau) \to \sigma \cap \alpha \to \tau$. Moreover the existence of both Id, Id^0 such that $\vdash$ Id$:\mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$ and $\vdash$ Id$^0:\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ does not imply their equality. Let us consider for example the types $\sigma = ((\tau \cap \rho \to \psi) \to \varphi) \cap ((\tau \to \psi) \cap \chi \to \varphi)$ and $\gamma = (\tau \cap \rho \to \psi) \to \varphi$. 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 $\vdash \lambda xy.x(\lambda z.yz) : \gamma \to \sigma$. We will discuss how to nd f.h.i.'s typed by $\mathcal{C}[\alpha\cap\sigma] \to \mathcal{C}[\sigma]$ and by $\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ in Section 7.

Observe that the path p$(\mathcal{C}[\ ])$ is unde ned if the hole is in an intersection with an atom: for example the path of $\sigma_1 \to [\ ] \cap (\sigma_2 \to \tau_1) \cap \varphi \to \tau_2$ is unde ned. Therefore we cannot reduce $\sigma_1 \to (\sigma_2 \to \rho_1 \cap \rho_2) \cap (\sigma_3 \to \tau_1) \cap \varphi \to \tau_2$ by splitting $\sigma_2 \to \rho_1 \cap \rho_2$. Similarly, 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 from providing the isomorphism between the original type and the simpli ed type: thus, while we have $(\sigma \cap \tau \to \rho) \cap (\sigma \to \rho) \rightsquigarrow \sigma \to \rho$, the type $(\sigma \cap \tau \to \rho) \cap (\sigma \to \rho) \cap \varphi$ does not reduce to $(\sigma \to \rho) \cap \varphi$. For any type $\sigma \equiv \bigcap_{i2I} \alpha_i$ such that $\alpha_i \not\equiv \alpha_j$ for all $i, j \in I$, the type $\sigma \cap \varphi$ (with φ atomic) is in normal form, since the atom φ blocks any reduction.

It is immediate to see that reduction by splitting and erasing is con uent and terminating, thus de ning a *type normal form*.

3.3 Similarity

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 3.6 *Similarity.* The *similarity* relation 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 $\langle\sigma_1^{(1)}\to\ldots\to\sigma_n^{(1)}\to\rho^{(1)},\ldots,\sigma_1^{(m)}\to\ldots\to\sigma_n^{(m)}\to\rho^{(m)}\rangle \sim$ $\langle\tau_{\pi(1)}^{(1)}\to\ldots\to\tau_{\pi(n)}^{(1)}\to\rho^{(1)},\ldots,\tau_{\pi(1)}^{(m)}\to\ldots\to\tau_{\pi(n)}^{(m)}\to\rho^{(m)}\rangle$, where π is a permutation of $1,\ldots,n$.

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

The reason is that, for two intersection types to be isomorphic, it is not su cient 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 expresses such property.

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 no 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

$$(\tau_1\to\tau_2\to\tau_3\to\rho_1)\cap(\sigma_1\to\sigma_2\to\sigma_3\to\rho_2),$$
$$(\tau_2\to\tau_3\to\tau_1\to\rho_1)\cap(\sigma_2\to\sigma_3\to\sigma_1\to\rho_2)$$

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

A type like $(\sigma_1\to\ldots\to\sigma_n\to\rho)\cap\varphi$ 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 $\sigma_1\to\ldots\to\sigma_n\to\rho$.

A more complex example of similar types is the following:

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

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:

$$\tau_1 = (\sigma_1 \cap \alpha \to \sigma_2 \to \rho_1) \cap (\sigma_3 \to \sigma_4 \to \rho_2),$$
$$\tau_2 = (\sigma_2 \to \sigma_1 \to \rho_1) \cap (\sigma_4 \to \alpha \cap \sigma_3 \to \rho_2).$$

They are not isomorphic, and are also not similar since the sequences $\langle \sigma_1 \cap \alpha, \sigma_3 \rangle$, $\langle \sigma_1, \alpha \cap \sigma_3 \rangle$ are not. If the definitions were given directly through intersection, owing to the associativity of $\cap$ the two sequences would be represented by the same intersection $\sigma_1 \cap \alpha \cap \sigma_3$, and the two types τ_1, τ_2 would be, incorrectly, considered similar.
An equivalent, slightly more algorithmic, definition of similarity may be given through a notion of *permutation tree*.

Definition 3.7 *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 an 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 an f.h.p. proper.

Definition 3.8 *F.h.p. corresponding to a permutation tree.*
The f.h.p. corresponding to a permutation tree Π is:

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

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

A permutation tree represents a tree of nested permutations: if weapply 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 de nition.

De nition 3.9 Application of a permutation tree to a type.
Application of a permutation tree is a partial map from types to types:

- $\varnothing(\sigma) = \sigma$
- $\langle \pi, [\Pi_1, \ldots, \Pi_n] \rangle (\sigma_1 \to \cdots \to \sigma_n \to \rho) = \Pi_1(\sigma_{\pi(1)}) \to \cdots \to \Pi_n(\sigma_{\pi(n)}) \to \rho$
- $\Pi(\sigma \cap \tau) = \Pi(\sigma) \cap \Pi(\tau)$
- $\Pi(\sigma)$ = unde ned 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

$$\tau = \gamma_2^0 \to \xi_3 \to \gamma_1^0 \to \xi$$

where

$$
\begin{aligned}
\gamma_2^0 &= \vartheta_3 \to \vartheta_1 \to (\psi_{42} \to \psi_{41} \to \vartheta_4) \to (\psi_{21} \to \psi_{23} \to \psi_{22} \to \vartheta_2) \to \vartheta \\
\gamma_1^0 &= (\varphi_{12} \to \varphi_{11} \to \chi_1) \to \chi_2 \to (\varphi_{33} \to \varphi_{32} \to \varphi_{31} \to \varphi_{34} \to \chi_3) \to \chi.
\end{aligned}
$$

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

De nition 3.10 Type permutation-equivalence
Two types σ and τ are permutation-equivalent, notation $\sigma \doteq \tau$, if $\exists \Pi . \Pi(\sigma) \equiv \tau$.

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

It is easy to prove that

PROPOSITION 3.11 SIMILARITY VS. PERMUTATION EQUIVALENCE. *For any types σ and τ, $\sigma \sim \tau$ if and only if $\sigma \doteq \tau$.*

So that the latter equivalence merely is an alternative de nition of the previously de ned similarity. We will therefore always use the rst notation.
As an immediate consequence of De nition 3.9, we have the following lemma.

LEMMA 3.12. *If $\Pi(\sigma) \equiv \tau$, with $\Pi = \langle \pi, [\Pi_1, \ldots, \Pi_n] \rangle$, then there exists a set I of indices such that σ and τ have the forms:*

$$\sigma \equiv \bigcap_{i2I}(\sigma_1^{(i)} \to \ldots \to \sigma_n^{(i)} \to \rho^{(i)}), \quad \tau \equiv \bigcap_{i2I}(\tau_1^{(i)} \to \ldots \to \tau_n^{(i)} \to \rho^{(i)})$$

and for all $i \in I$, for $k = 1, \ldots, n$, one has $\Pi_k(\sigma_{\pi(k)}^{(i)}) \equiv \tau_k^{(i)}$, therefore $\sigma_{\pi(k)}^{(i)} \sim \tau_k^{(i)}$.

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

$$\bigcap_{i2I}(\sigma_1^{(i)} \to \ldots \to \sigma_n^{(i)} \to \rho^{(i)}) \ \sim \ \bigcap_{i2I}(\tau_1^{(i)} \to \ldots \to \tau_n^{(i)} \to \rho^{(i)})$$

if there exists a permutation π such that

$$\forall i \in I \,.\, \tau_k^{(i)} \sim \sigma_{\pi(k)}^{(i)} \quad \text{and} \quad \bigcap_{i2I} \tau_k^{(i)} \sim \bigcap_{i2I} \sigma_{\pi(k)}^{(i)} \quad \text{for } k = 1, \ldots, n.$$

A counterexample is given by the following pair of types:

$$\begin{aligned} \sigma &= (\beta_1 \to \alpha_1) \cap (\beta_2 \to \alpha_2) \cap (\beta_3 \to \alpha_3) \\ \tau &= (\gamma_1 \to \alpha_1) \cap (\gamma_2 \to \alpha_2) \cap (\gamma_3 \to \alpha_3) \end{aligned}$$

where

$$\begin{aligned} \beta_1 &= \varphi \to \chi \to \psi \to \vartheta = \gamma_2 \\ \beta_2 &= \varphi \to \psi \to \chi \to \vartheta = \gamma_3 \\ \beta_3 &= \chi \to \varphi \to \psi \to \vartheta = \gamma_1. \end{aligned}$$

We have $\Pi_1(\beta_1) \equiv \gamma_1, \quad \Pi_2(\beta_2) \equiv \gamma_2, \quad \Pi_3(\beta_3) \equiv \gamma_3$, 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 rst de nition 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 6.5, not isomorphic ($\sigma \not\approx \tau$).

4. STANDARD PROPERTIES OF THE TYPE SYSTEM

Since our system is a minor variant of the standard system [Barendregt et al. 1983], it has the usual properties: in particular a generation lemma holds, the subject reduction and the subject expansion properties hold for terms belonging to the $\lambda\mathbf{I}$-calculus, and the proofs are routine.

Lemma 4.1 Generation Lemma.

(*1*) *If* $x : \bigcap_{i \in I} \mu_i \vdash x : \bigcap_{j \in J} \nu_j$*, then* $\{\nu_j \mid j \in J\} \subseteq \{\mu_i \mid i \in I\}$.

(*2*) *If* $\vdash \lambda x.M : \bigcap_{i \in I}(\sigma_i \to \tau_i)$*, then for all* $i \in I$*:* $, x : \sigma_i \vdash M : \tau_i$.

(*3*) *If* $\vdash MN : \tau$*, then there exist types* $\sigma_i, \tau_i (i \in I)$ *such that* $\vdash M : \bigcap_{i \in I}(\sigma_i \to \tau_i)$ *and* $\vdash N : \bigcap_{i \in I} \sigma_i$ *and either* $\tau = \bigcap_{i \in I} \tau_i$ *or* $\tau \cap \rho = \bigcap_{i \in I} \tau_i$ *for some* ρ.

(*4*) *If* $\vdash MN : \mu$*, then there exists a type* σ *such that* $\vdash N : \sigma$ *and either* $\vdash M : \sigma \to \mu$ *or* $\vdash M : \sigma \to \mu \cap \rho$ *for some* ρ.

Proof. The proof is by induction on derivations. Point (4) easily follows from Point (3). □

Note that ρ is needed in Points (3) and (4), for example $x : \sigma \to \tau \cap \rho, y : \sigma \vdash xy : \tau$.

Theorem 4.2 Subject Reduction.
If $\vdash M : \sigma$ *and* $M \longrightarrow_\beta N$*, then* $\vdash N : \sigma$.

Proof. Standard. □

Theorem 4.3 Subject Expansion for $\lambda\mathbf{I}$-terms.
If $\vdash M : \sigma$*,* N *is a* $\lambda\mathbf{I}$*-term, and* $N \longrightarrow_\beta M$*, then* $\vdash N : \sigma$.

Proof. Standard. □

Lemmata 4.5 and 4.8 state some useful properties of η-expansions of the identity and of permutators. In particular, Lemma 4.5.1 says that an f.h.i. is able to map an intersection $\sigma \cap \tau$ to one of its components, for example σ, only if it is able to map such component σ to itself (which is not always the case, since the number of top-level arrows in σ cannot be less than the number of top-level abstractions of the f.h.i.). Lemma 4.5.2 states the rather obvious fact that if an 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.

A key notion for characterising which types can be inhabited by f.h.p.'s is the minimal number of top arrows, as defined in Definition 4.6. Lemma 4.7 relates the number of top arrows to the agreement with paths (Point (1)), to the shapes of arrows in types which cannot be reduced by the splitting rule (Point (2)) and to derivability for λ-abstractions (Point (3)).

Finally, Lemma 4.8 states that if an f.h.p. P maps an intersection $\bigcap_{i \in I} \mu_i$ to another intersection $\bigcap_{j \in J} \nu_j$, i.e., $\vdash$ P: $\bigcap_{i \in I} \mu_i \to \bigcap_{j \in J} \nu_j$ - under the hypothesis that the number of top arrows of $\bigcap_{i \in I} \mu_i$ is not lower than the number of top arrows of $\bigcap_{j \in J} \nu_j$, and that all types are in normal form w.r.t. the splitting rule - then every component ν_j in the target intersection is obtained by P from some component μ_i in the source intersection (Point (1)). Moreover this lemma gives other properties of the types and of the typing of P subterms (Point (2)).

Remark 4.4. In the following we write judgments of the form $x : \sigma \vdash$ P$x : \tau$ (where P may also be Id) instead of $\vdash$ P: $\sigma \to \tau$, in order to simplify the proofs. The two kinds of judgments can be easily shown equivalent as follows. Because an f.h.p. P is an abstraction we can assume P = $\lambda x.M$. From $x : \sigma \vdash$ P$x : \tau$ by β-reduction and subject reduction one has $x : \sigma \vdash M : \tau$, whence, by ($\to$ I), $\vdash$ P: $\sigma \to \tau$. The opposite implication, from $\vdash$ P: $\sigma \to \tau$ to $x : \sigma \vdash$ P$x : \tau$, trivially follows by ($\to$ E).

LEMMA 4.5. (*1*) *If* $x{:}\sigma\cap\tau \vdash \mathsf{Id}\,x{:}\sigma$, *then* $x{:}\sigma \vdash \mathsf{Id}\,x{:}\sigma$.

(*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. □

Definition 4.6. The minimum number of top arrows in a type τ is noted $\#(\tau)$ and is defined inductively as:

$$\#(\varphi) = 0 \qquad \#(\sigma\to\tau) = 1 + \#(\tau) \qquad \#(\sigma\cap\tau) = \min(\#(\sigma), \#(\tau)).$$

For example $\#(\sigma_1 \to \sigma_2 \to \tau\cap\varphi) = 1 + \#(\sigma_2 \to \tau\cap\varphi) = 2 + \#(\tau\cap\varphi) = 2 + \min(\#(\tau), \#(\varphi)) = 2$.

LEMMA 4.7. (*1*) $\sigma \propto \langle n\rangle$ *if and only if* $\#(\sigma) \geq n$, *for all* $n \geq 1$.

(*2*) *If* $\#(\bigcap_{i\in I}\mu_i) \geq n$ *and* $\bigcap_{i\in I}\mu_i$ *does not contain subtypes which can be split, then for all* $i\in I$ *there are* $\sigma_1^{(i)},\ldots,\sigma_n^{(i)},\nu^{(i)}$ *such that* $\mu_i = \sigma_1^{(i)}\to\ldots\to\sigma_n^{(i)}\to\nu^{(i)}$.

(*3*) *If* $\vdash \lambda x_1\ldots x_n.M{:}\sigma$, *then* $\#(\sigma)\geq n$.

PROOF. Points (1) and (3) can be easily shown by induction on n. For (2) assume towards a contradiction that for some $j\in I$ and $m \leq n$ we get $\mu_j = \sigma_1^{(j)} \to \ldots \to \sigma_m^{(j)} \to \tau\cap\rho$. Note that m cannot be 0 being μ_j an atomic or an arrow type. For $m\geq 1$ point (1) implies $\bigcap_{i\in I, i\neq j}\mu_i \propto \langle m\rangle$ and then by definition $\mathsf{p}((\sigma_1^{(j)} \to \ldots \to \sigma_m^{(j)} \to [\ ]) \cap \bigcap_{i\in I, i\neq j}\mu_i) = \langle m\rangle$ is defined. We conclude by Definition 3.3 that $\bigcap_{i\in I}\mu_i$ contains a subtype which can be split. □

LEMMA 4.8. *Let* $\bigcap_{i\in I}\mu_i$, $\bigcap_{j\in J}\nu_j$ *do not contain subtypes which can be split and* $\#(\bigcap_{i\in I}\mu_i) \geq \#(\bigcap_{j\in J}\nu_j)$. *Then* $\mathsf{P}\ {}_{\beta}\!\longleftarrow \lambda yz_1\ldots z_n.y(\mathsf{P}_1 z_{\pi(1)})\ldots(\mathsf{P}_n z_{\pi(n)})$ *and* $x{:}\bigcap_{i\in I}\mu_i \vdash \mathsf{P}x{:}\bigcap_{j\in J}\nu_j$ *imply* $\forall j\in J.\ \exists i_j\in I$ *such that:*

(*1*) $x{:}\mu_{i_j} \vdash \mathsf{P}x{:}\nu_j$ *and*

(*2*) $\mu_{i_j} = \tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)}$, $\nu_j = \sigma_1^{(j)} \to \ldots \to \sigma_n^{(j)} \to \lambda^{(j)}$, *and* $z_{\pi(l)}{:}\sigma_{\pi(l)}^{(j)} \vdash \mathsf{P}_l z_{\pi(l)}{:}\tau_l^{(j)}$ ($1 \leq l \leq n$) *for some* $\tau_1^{(j)},\ldots,\tau_n^{(j)},\sigma_1^{(j)},\ldots,\sigma_n^{(j)},\lambda^{(j)}$.

PROOF. See Table II, where $\mathsf{P}^0 = \lambda z_1\ldots z_n.x(\mathsf{P}_1 z_{\pi(1)})\ldots(\mathsf{P}_n z_{\pi(n)})$. □

It is easy to see that $\forall j\in J.\ \exists i_j\in I.\ x{:}\mu_{i_j} \vdash \mathsf{P}x{:}\nu_j$ implies $x{:}\bigcap_{i\in I}\mu_i \vdash \mathsf{P}x{:}\bigcap_{j\in J}\nu_j$ by application of the ($\cap E$) rule and then of the ($\cap I$) rule.

5. SOUNDNESS OF THE REDUCTION BY SPLITTING

To each path we can naturally associate an f.h.i. (Definition 5.1) which maps to itself each type which agrees with the path (Lemma 5.2).

Definition 5.1. The f.h.i. induced by the path p (notation Id_{p}) is defined by induction on p:

- $\mathsf{Id}_{\langle\rangle} = \lambda y.y$;
- $\mathsf{Id}_{\langle 1,n_2,\ldots,n_m\rangle}\ {}_{\beta}\!\longleftarrow \lambda yz.y(\mathsf{Id}_{\langle n_2,\ldots,n_m\rangle}z)$;
- $\mathsf{Id}_{\langle n+1,n_2,\ldots,n_m\rangle}\ {}_{\beta}\!\longleftarrow \lambda yz.\mathsf{Id}_{\langle n,n_2,\ldots,n_m\rangle}(yz)$.

Table II Proof of Lemma 4.8

$x:\bigcap_{i\in I}\mu_i \vdash \mathsf{P}x:\bigcap_{j\in J}\nu_j$	$\Longrightarrow$	$x:\bigcap_{i\in I}\mu_i \vdash \mathsf{P}':\bigcap_{j\in J}\nu_j$
		by Theorems 4.2 and 4.3
	$\Longrightarrow$	$\forall j \in J.\ x:\bigcap_{i\in I}\mu_i \vdash \mathsf{P}':\nu_j$
		by rule $(\cap E)$
	$\Longrightarrow$	$\forall j \in J.\ \#(\nu_j) \geq n$ by Lemma 4.7(3)
	$\Longrightarrow$	$\forall j \in J.\ \nu_j = \sigma_1^{(j)} \to \ldots \to \sigma_n^{(j)} \to \lambda^{(j)}$
		for some $\sigma_1^{(j)}, \ldots, \sigma_n^{(j)}, \lambda^{(j)}$
		by Lemma 4.7(2)
	$\Longrightarrow$	$\forall j \in J.\quad \vdash x(\mathsf{P}_1 z_{\pi(1)})\ldots(\mathsf{P}_n z_{\pi(n)}):\lambda^{(j)}$
		where $\quad = x:\bigcap_{i\in I}\mu_i, z_1:\sigma_1^{(j)}, \ldots z_n:\sigma_n^{(j)}$
		by Lemma 4.1(2)
	$\Longrightarrow$	$\forall j \in J.\ z_{\pi(1)}:\sigma_{\pi(1)}^{(j)} \vdash \mathsf{P}_1 z_{\pi(1)}:\tau_1^{(j)}$ & $\ldots$
		& $z_{\pi(n)}:\sigma_{\pi(n)}^{(j)} \vdash \mathsf{P}_n z_{\pi(n)}:\tau_n^{(j)}$
		for some $\tau_1^{(j)}, \ldots, \tau_n^{(j)}$ and
		either $x:\bigcap_{i\in I}\mu_i \vdash x:\tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)}$
		or $x:\bigcap_{i\in I}\mu_i \vdash x:\tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)} \cap \rho^{(j)}$
		for some $\rho^{(j)}$
		by Lemma 4.1(4)
	$\Longrightarrow$	$\forall j \in J.\ \exists i_j \in I.\ \mu_{i_j} = \tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)}$
		or $\mu_{i_j} = \tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)} \cap \rho^{(j)}$
		by Lemma 4.1(1)
	$\Longrightarrow$	$\forall j \in J.\ \exists i_j \in I.\ \mu_{i_j} = \tau_1^{(j)} \to \ldots \to \tau_n^{(j)} \to \lambda^{(j)}$
		by Lemma 4.7(2) since $\bigcap_{i\in I}\mu_i$
		does not contain subtypes which can be split
		and $\#(\bigcap_{i\in I}\mu_i) \geq \#(\bigcap_{j\in J}\nu_j) \geq n$
	$\Longrightarrow$	$\forall j \in J.\ \exists i_j \in I.\ x:\mu_{i_j} \vdash \mathsf{P}x:\nu_j$
		by rules $(\to E)$ and $(\to I)$.

For example $\mathsf{Id}_{\langle 2,1\rangle}\ {}_\beta\!\longleftarrow\ \lambda y_1 z_1.\mathsf{Id}_{\langle 1,1\rangle}(y_1 z_1)\ \ {}_\beta\!\longleftarrow\ \lambda y_1 z_1.(\lambda y_2 z_2.y_2(\mathsf{Id}_{\langle 1\rangle} z_2))(y_1 z_1)$ ${}_\beta\!\longleftarrow\ \lambda y_1 z_1.(\lambda y_2 z_2.y_2((\lambda y_3 z_3.y_3(\mathsf{Id}_{\langle\rangle} z_3))z_2))(y_1 z_1)$, which in turn expands to the term $\lambda y_1 z_1.(\lambda y_2 z_2.y_2((\lambda y_3 z_3.y_3((\lambda y_4.y_4)z_3))z_2))(y_1 z_1)$, so we can conclude that $\mathsf{Id}_{\langle 2,1\rangle} = \lambda y_1 z_1 z_2.y_1 z_1(\lambda z_3.z_2 z_3)$.

LEMMA 5.2. *If* $\sigma \propto \mathsf{p}$, *then* $\vdash \mathsf{Id}_{\mathsf{p}}:\sigma \to \sigma$.

PROOF. By induction on σ and p. If $\sigma = \tau \to \rho$ and $\mathsf{p} = \langle 1, n_2, \ldots, n_m\rangle$, then by induction $\vdash \mathsf{Id}_{\langle n_2,\ldots,n_m\rangle}:\tau \to \tau$, which implies $\vdash \lambda yz.y(\mathsf{Id}_{\langle n_2,\ldots,n_m\rangle} z):\sigma \to \sigma$. If $\sigma = \tau \to \rho$ and $\mathsf{p} = \langle n+1, n_2, \ldots, n_m\rangle$, then by induction $\vdash \mathsf{Id}_{\langle n,n_2,\ldots,n_m\rangle}:\rho \to \rho$, which implies $\vdash \lambda yz.\mathsf{Id}_{\langle n,n_2,\ldots,n_m\rangle}(yz):\sigma \to \sigma$. If $\sigma = \tau \cap \rho$, then by definition $\tau \propto \mathsf{p}$ and $\rho \propto \mathsf{p}$. This case easily follows by induction using Lemma 4.5(2). $\square$

We are now able to show the soundness of the reduction by splitting by using the f.h.i. associated to the path $\mathsf{p}(\mathcal{C}[\,])$.

THEOREM 5.3.
If $\mathsf{p}(\mathcal{C}[\,])$ *is defined, then* $\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}[\,])} : \mathcal{C}[\sigma \to \tau \cap \rho] \to \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)]$ *and* $\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}[\,])} : \mathcal{C}[(\sigma \to \tau) \cap (\sigma \to \rho)] \to \mathcal{C}[\sigma \to \tau \cap \rho]$ *for arbitrary* σ*,* τ*,* ρ*.*

PROOF. By induction on $\mathcal{C}[\,]$. The case $\mathcal{C}[\,] = [\,]$ is easy, since by de nition $\mathsf{p}([\,]) = \langle 1 \rangle$ and $\mathsf{Id}_{h1i} = \lambda yz.yz$.

If $\mathcal{C}[\,] = \mathcal{C}'[\,] \to \sigma'$, then by induction

$$\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} : \mathcal{C}'[\sigma \to \tau \cap \rho] \to \mathcal{C}'[(\sigma \to \tau) \cap (\sigma \to \rho)]$$
$$\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} : \mathcal{C}'[(\sigma \to \tau) \cap (\sigma \to \rho)] \to \mathcal{C}'[\sigma \to \tau \cap \rho].$$

Since by de nition $\mathsf{Id}_{\mathsf{p}(\mathcal{C}[\,])} \;{}_{\beta}\!\longleftarrow \lambda yz.y(\mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} z)$ the result follows.

If $\mathcal{C}[\,] = \sigma' \to \mathcal{C}'[\,]$ the proof is similar to that of previous case.

If $\mathcal{C}[\,] = \mathcal{C}'[\,] \cap \sigma'$, then by induction

$$\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} : \mathcal{C}'[\sigma \to \tau \cap \rho] \to \mathcal{C}'[(\sigma \to \tau) \cap (\sigma \to \rho)]$$
$$\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} : \mathcal{C}'[(\sigma \to \tau) \cap (\sigma \to \rho)] \to \mathcal{C}'[\sigma \to \tau \cap \rho].$$

By de nition $\mathsf{p}(\mathcal{C}[\,]) = \mathsf{p}(\mathcal{C}'[\,])$ and $\sigma' \propto \mathsf{p}(\mathcal{C}'[\,])$. From Lemma 5.2 we have $\vdash \mathsf{Id}_{\mathsf{p}(\mathcal{C}'[\,])} : \sigma' \to \sigma'$ and so we conclude by Lemma 4.5(2). $\square$

6. 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.

The rst result is that isomorphic types have the same number of top arrows.

LEMMA 6.1. *If* $\#(\sigma) \neq \#(\tau)$*, then* σ *and* τ *cannot be isomorphic.*

PROOF. Since reduction by splitting does not change the number of top arrows, we can assume without loss of generality that σ and τ do not contain subtypes which can be split. If $n = \#(\sigma) < \#(\tau)$, then $\sigma = (\sigma_1 \to \ldots \to \sigma_n \to \varphi) \cap \sigma'$ and $\tau = \bigcap_{i2I}(\tau_1^{(i)} \to \ldots \to \tau_{n+1}^{(i)} \to \rho^{(i)})$ for suitable $\sigma_1, \ldots, \sigma_n, \varphi, \sigma', \tau_1^{(i)}, \ldots, \tau_{n+1}^{(i)}, \rho^{(i)}$ $(i \in I)$, by the shape of normal forms w.r.t. the splitting rule and by de nition of $\#$. Let's assume, towards a contradiction, that $x{:}\tau \vdash \mathsf{P}x{:}\sigma$, where $\mathsf{P} \;{}_{\beta}\!\longleftarrow \lambda yz_1 \ldots z_m.y(\mathsf{P}_1 z_{\pi(1)}) \ldots (\mathsf{P}_m z_{\pi(m)})$. By Lemma 4.8(2) there is $j \in I$ such that

$$\tau_1^{(j)} \to \ldots \to \tau_{n+1}^{(j)} \to \rho^{(j)} = \sigma_1^{(j)} \to \ldots \to \sigma_m^{(j)} \to \sigma_{m+1} \to \ldots \to \sigma_n \to \varphi$$

for some $\sigma_1^{(j)}, \ldots, \sigma_m^{(j)}$, i.e., $\tau_{n+1}^{(j)} \to \rho^{(j)} = \varphi$, which is impossible. $\square$

If we only consider normal types, we can strengthen Lemma 4.8(1) by Lemma 6.3, which states that if an f.h.p. P has the type $\bigcap_{i2I}\mu_i \to \bigcap_{j2J}\nu_j$ and $\bigcap_{j2J}\nu_j$ has no more top arrows than $\bigcap_{i2I}\mu_i$, then not only $\forall j{\in}J\,.\,\exists i_j \in I.\vdash \mathsf{P}{:}\,\mu_{i_j} \to \nu_j$, but its inverse $\mathsf{P}\;\;^1$ precisely maps each component ν_j of the target intersection to its corresponding μ_{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 6.2 is instrumental to the proof of Lemma 6.3, 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 by starting from the others.

LEMMA 6.2. *If $\tau \cap \mu$ is normal, then there is no* Id *such that* $x{:}\tau \vdash \mathsf{Id}\,x{:}\tau\cap\mu$.

PROOF. Let $\tau = \bigcap_{i\in I}\mu_i$. Towards a contradiction assume $x:\tau \vdash \mathsf{Id}\,x:\tau\cap\mu$. Since $x{:}\tau\cap\mu\vdash x{:}\tau$ we get $\tau\cap\mu \leadsto \tau$. $\square$

LEMMA 6.3. *If $\bigcap_{i\in I}\mu_i, \bigcap_{j\in J}\nu_j$ are normal types, and $\#(\bigcap_{i\in I}\mu_i) \geq \#(\bigcap_{j\in J}\nu_j)$, and $x{:}\bigcap_{i\in I}\mu_i \vdash \mathsf{P}x{:}\bigcap_{j\in J}\nu_j$, and $x{:}\bigcap_{j\in J}\nu_j \vdash \mathsf{P}^{-1}x{:}\bigcap_{i\in I}\mu_i$, and $x{:}\mu_{i_0}\vdash \mathsf{P}x{:}\nu_{j_0}$, then $x{:}\nu_{j_0}\vdash \mathsf{P}^{-1}x{:}\mu_{i_0}$.*

PROOF. By Lemma 4.8(1) there is $j_1\in J$ such that $x{:}\nu_{j_1}\vdash \mathsf{P}^{-1}x{:}\mu_{i_0}$. We assume $j_0 \neq j_1$ towards a contradiction. From $x:\nu_{j_1}\vdash \mathsf{P}^{-1}x:\mu_{i_0}$ and $x:\mu_{i_0}\vdash \mathsf{P}x:\nu_{j_0}$ we get $x:\nu_{j_1}\vdash \mathsf{P}(\mathsf{P}^{-1}x):\nu_{j_0}$, which implies $x:\bigcap_{j\in J, j\neq j_0}\nu_j \vdash (\mathsf{P}\circ\mathsf{P}^{-1})x:\bigcap_{j\in J}\nu_j$ by Lemma 4.5. This is, by Lemma 6.2, impossible, since $\mathsf{P}\circ\mathsf{P}^{-1}$ is β-reducible to an f.h.i. $\square$

THEOREM 6.4 SOUNDNESS OF $\sim$. *If σ and τ are arbitrary types, then $\sigma\sim\tau$ implies $\sigma\approx\tau$.*

PROOF. We show that $\langle\sigma_1,\ldots,\sigma_m\rangle \sim \langle\tau_1,\ldots,\tau_m\rangle$ implies that there is an f.h.p. P such that $\vdash \mathsf{P}{:}\sigma_j\to\tau_j$ for $1\leq j\leq m$, proceeding by induction on the definition of $\sim$. The only interesting case is

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

with $\langle\sigma_i^{(1)},\ldots,\sigma_i^{(m)}\rangle\sim\langle\tau_i^{(1)},\ldots,\tau_i^{(m)}\rangle$ for $1\leq i\leq n$.
By induction, there is a P_i such that $\vdash \mathsf{P}_i{:}\sigma_i^{(j)}\to\tau_i^{(j)}$ for $1\leq j\leq m$. We can then choose P as the β-normal form of $\lambda yz_1\ldots z_n.y(\mathsf{P}_1 z_{\pi^{-1}(1)})\ldots(\mathsf{P}_n z_{\pi^{-1}(n)})$. $\square$

The opposite implication does not hold: two isomorphic types are not necessarily similar. The simplest example is $\sigma\to\tau\cap\rho$ and $(\sigma\to\tau)\cap(\sigma\to\rho)$. Also isomorphic types not containing subtypes that can be split may be not similar. For instance, the type $\sigma = ((\tau\cap\rho\to\psi)\to\varphi)\cap((\tau\to\psi)\cap\chi\to\varphi)$ and its normal form $\gamma = (\tau\cap\rho\to\psi)\to\varphi$, 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 6.5 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 6.4).

We show by structural induction on P that if $\vdash \mathsf{P}:\sigma_j\to\tau_j$ and $\vdash \mathsf{P}^{-1}:\tau_j\to\sigma_j$ for $1\leq j\leq m$, then $\langle\sigma_1,\ldots,\sigma_m\rangle\sim\langle\tau_1,\ldots,\tau_m\rangle$. By Lemma 6.1 $\#(\sigma_j) = \#(\tau_j)$. Let $\sigma_j = \bigcap_{1\leq i\leq n_j}\mu_i^{(j)}$ and $\tau_j = \bigcap_{1\leq i\leq p_j}\nu_i^{(j)}$.
By Lemma 6.3 we get $n_j = p_j$ and $\vdash \mathsf{P}:\mu_i^{(j)}\to\nu_i^{(j)}$ and $\vdash \mathsf{P}^{-1}:\nu_i^{(j)}\to\mu_i^{(j)}$. Let

P $_\beta\longleftarrow \lambda y z_1 \dots z_n . y(\mathsf{P}_1 z_{\pi(1)}) \dots (\mathsf{P}_n z_{\pi(n)})$. By Lemma 4.8(2), we get

$$\mu_i^{(j)} = \tau_1^{(i,j)} \to \dots \to \tau_n^{(i,j)} \to \lambda^{(i,j)} \text{ and } \nu_i^{(j)} = \sigma_1^{(i,j)} \to \dots \to \sigma_n^{(i,j)} \to \lambda^{(i,j)}$$

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

$$\langle \sigma_{\pi(l)}^{(1,1)}, \dots, \sigma_{\pi(l)}^{(n_1,1)}, \dots, \sigma_{\pi(l)}^{(1,m)}, \dots, \sigma_{\pi(l)}^{(n_m,m)} \rangle \sim \langle \tau_l^{(1,1)}, \dots, \tau_l^{(n_1,1)}, \dots, \tau_l^{(1,m)}, \dots, \tau_l^{(n_m,m)} \rangle$$

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

$$\langle \mu_1^{(1)}, \dots, \mu_{n_1}^{(1)}, \dots, \mu_1^{(m)}, \dots, \mu_{n_m}^{(m)} \rangle \sim \langle \nu_1^{(1)}, \dots, \nu_{n_1}^{(1)}, \dots, \nu_1^{(m)}, \dots, \nu_{n_m}^{(m)} \rangle$$

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

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

THEOREM 6.6. *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., $\sigma \approx \tau$.

□

7. HOW TO NORMALISE TYPES

The application of the type reduction rule, as de ned in Section 3, su ers from combinatorial explosion in the search for the splittable and erasable type subexpression α, thus possibly making the normalisation impractical. However, the search space can be considerably reduced with a more accurate formulation of the algorithm. Thanks to con uence we can rst apply splitting and then erasure. So we will only consider types that are in normal form w.r.t. the splitting rule.

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 \→" as a function space constructor and of \∩" as set intersection:

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

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

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

$$\sigma^0 \leq \sigma,\ \tau \leq \tau^0 \Rightarrow \sigma \to \tau \leq \sigma^0 \to \tau^0.$$

Note that we do not need the axiom $(\sigma \to \tau) \cap (\sigma \to \rho) \leq \sigma \to \tau \cap \rho$, since we assume that the current types are irreducible w.r.t. the splitting rule. It is easy to verify that an algorithmic equivalent definition of $\leq$ is:

$$\bigcap_{i \in I}(\sigma_i \to \tau_i) \cap \bigcap_{h \in H} \varphi_h \leq \bigcap_{j \in J}(\sigma'_j \to \tau'_j) \cap \bigcap_{k \in K} \varphi'_k$$

$$\text{if } \forall j \in J \exists i \in I. \sigma'_j \leq \sigma_i \ \& \ \tau_i \leq \tau_j \text{ and } \forall k \in K \exists h \in H. \varphi_h = \varphi'_k.$$

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 there exist two f.h.i.'s Id, Id' with suitable types, i.e. such that $\vdash \mathsf{Id} : \mathcal{C}[\beta] \to \mathcal{C}[\alpha \cap \beta]$ and $\vdash \mathsf{Id}' : \mathcal{C}[\alpha \cap \beta] \to \mathcal{C}[\beta]$, where $\sigma = \mathcal{C}[\alpha \cap \beta]$. This can be performed through the following mapping $\mathcal{I}$ which, applied to two types σ and τ, builds the set of all f.h.i.'s Id such that $\vdash \mathsf{Id} : \sigma \to \tau$.

$$\begin{aligned}
\mathcal{I}(\varphi, \varphi') &= \varnothing \quad \text{if } \varphi \neq \varphi' \\
\mathcal{I}(\varphi, \sigma \to \tau) &= \mathcal{I}(\sigma \to \tau, \varphi') = \varnothing \\
\mathcal{I}(\varphi, \varphi) &= \{\lambda x.x\}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{I}(\sigma \to \tau, \sigma \to \tau) \\
&\quad = \{\lambda x.x\} \\
&\qquad \cup \{\mathsf{Id} \mid \lambda yz.\mathsf{Id}_1(y(\mathsf{Id}_2 z)) \longrightarrow_\beta \mathsf{Id} \ \& \ \mathsf{Id}_1 \in \mathcal{I}(\tau, \tau) \ \& \ \mathsf{Id}_2 \in \mathcal{I}(\sigma, \sigma)\} \\
&\mathcal{I}(\sigma \to \tau, \sigma' \to \tau') \\
&\quad = \{\mathsf{Id} \mid \lambda yz.\mathsf{Id}_1(y(\mathsf{Id}_2 z)) \longrightarrow_\beta \mathsf{Id} \ \& \ \mathsf{Id}_1 \in \mathcal{I}(\tau, \tau') \ \& \ \mathsf{Id}_2 \in \mathcal{I}(\sigma', \sigma)\} \\
&\qquad \text{if } \sigma \neq \sigma' \text{ or } \tau \neq \tau' \\
&\mathcal{I}(\bigcap_{i \in I} \mu_i, \bigcap_{j \in J} \nu_j) \\
&\quad = \{\mathsf{Id} \mid \forall j \in J \ \exists i \in I. \ \mathsf{Id} \in \mathcal{I}(\mu_i, \nu_j)\}.
\end{aligned}$$

The correctness of the last clause defining the mapping $\mathcal{I}$ follows from Lemma 4.8(1), which can be applied since the current types cannot be split and since by Lemma 6.1 two isomorphic types must have the same number of top arrows. The correctness of the other clauses defining the mapping $\mathcal{I}$ follows from the following lemma, which can be easily proved using the Generation Lemma.

LEMMA 7.1. (*1*) *If* $\varphi \neq \varphi'$, *then there is no* Id *such that* $\vdash \mathsf{Id} : \varphi \to \varphi'$.

(*2*) *There is no* Id *such that* $\vdash \mathsf{Id} : \varphi \to \sigma \to \tau$ *or* $\vdash \mathsf{Id} : (\sigma \to \tau) \to \varphi$.

(*3*) *If* $\vdash \mathsf{Id} : (\sigma \to \tau) \to \sigma' \to \tau'$ *and* $\sigma \neq \sigma'$ *or* $\tau \neq \tau'$, *then* $\mathsf{Id} \ {}_\beta\!\longleftarrow \lambda yz.\mathsf{Id}_1(y(\mathsf{Id}_2 z))$ *for some* $\mathsf{Id}_1, \mathsf{Id}_2$ *such that* $\vdash \mathsf{Id}_1 : \tau \to \tau'$ *and* $\vdash \mathsf{Id}_2 : \sigma' \to \sigma$.

We end this session by a property of f.h.i.'s that allows us to search for only one between Id and Id' such that $\vdash \mathsf{Id} : \mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ and $\vdash \mathsf{Id}' : \mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$. For this we need a type preservation under η-reduction for f.h.i's.

LEMMA 7.2. *If* $x : \sigma \vdash \mathsf{Id} x : \sigma$ *and* $\mathsf{Id} \longrightarrow_\eta \mathsf{Id}'$, *then* $x : \sigma \vdash \mathsf{Id}' x : \sigma$.

PROOF. Standard by induction on $\longrightarrow_\eta$, using Lemma 4.1 and assuming $\mathsf{Id}\ {}_\beta\!\longleftarrow \lambda y z_1 \ldots z_n.y(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n)$ for some $n \geq 1$. $\square$

We de ne *positive* and *negative* occurrences of holes in contexts as expected:

- the occurrence of the hole in [ ] is positive,
- if the occurrence of the hole is positive (respectively negative) in $\mathcal{C}[\ ]$ then
  - it is positive (respectively negative) in $\rho \to \mathcal{C}[\ ]$ and in $\rho \cap \mathcal{C}[\ ]$,
  - it is negative (respectively positive) in $\mathcal{C}[\ ] \to \rho$.

It is easy to verify that when the occurrence of the hole is positive in $\mathcal{C}[\ ]$, then $\mathcal{C}[\alpha \cap \sigma] \leq \mathcal{C}[\sigma]$, and vice versa if the occurrence of the hole is negative in $\mathcal{C}[\ ]$, then $\mathcal{C}[\sigma] \leq \mathcal{C}[\alpha \cap \sigma]$. Clearly this is due to the contra-variance and the co-variance of the arrow type constructor.

The subtyping $\mathcal{C}[\alpha \cap \sigma] \leq \mathcal{C}[\sigma]$ suggests that we can nd an f.h.i. which inhabits $\mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$ as soon as we can nd an f.h.i. which \reaches" the hole in $\mathcal{C}[\ ]$. This can be assured by the existence of an f.h.i. which inhabits $\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ but does not inhabit $\mathcal{C}[\varphi] \to \mathcal{C}[\alpha \cap \sigma]$ when φ does not occur in $\mathcal{C}[\alpha \cap \sigma]$. Similarly when the hole occurrence is negative. This intuition is formalised in the following lemma.

LEMMA 7.3. *Let* $\#(\mathcal{C}[\sigma]) = \#(\mathcal{C}[\alpha \cap \sigma])$ *hold, and let* $\mathcal{C}[\alpha \cap \sigma]$ *do not contain subtypes which can be split.*

(1) *If* $\vdash \mathsf{Id}:\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ *and* $\not\vdash \mathsf{Id}:\mathcal{C}[\varphi] \to \mathcal{C}[\alpha \cap \sigma]$, *where* φ *does not occur in* $\mathcal{C}[\alpha\cap\sigma]$ *and the hole occurrence is positive, then there is* Id' *such that* $\mathsf{Id} \longrightarrow_\eta \mathsf{Id}'$ *and* $\vdash \mathsf{Id}':\mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$.

(2) *If* $\vdash \mathsf{Id}:\mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$ *and* $\not\vdash \mathsf{Id}:\mathcal{C}[\alpha \cap \varphi] \to \mathcal{C}[\sigma]$, *where* φ *does not occur in* $\mathcal{C}[\alpha\cap\sigma]$ *and the hole occurrence is negative, then there is* Id' *such that* $\mathsf{Id} \longrightarrow_\eta \mathsf{Id}'$ *and* $\vdash \mathsf{Id}':\mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$.

PROOF. We show both points simultaneously by induction on $\mathcal{C}[\ ]$.

First steps. If $\mathcal{C}[\ ] = [\ ]$, then $\vdash \lambda x.x : \alpha \cap \sigma \to \sigma$. If $\mathcal{C}[\ ] = [\ ] \to \rho$, then $\vdash \lambda xy.xy:(\sigma \to \rho) \to \alpha \cap \sigma \to \rho$.

Induction steps. We always assume $\mathsf{Id}\ {}_\beta\!\longleftarrow \lambda xyz_1 \ldots z_n.x(\mathsf{Id}_0 y)(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n)$ for some $n \geq 0$. We only consider the cases in which the hole occurrence is positive in $\mathcal{C}[\ ]$, since the proof is similar when the hole occurrence in $\mathcal{C}[\ ]$ is negative.

If $\mathcal{C}[\ ] = \mathcal{C}'[\ ] \to \rho$, then by Lemma 4.1(2)

$$x:\mathcal{C}'[\sigma] \to \rho, y:\mathcal{C}'[\alpha \cap \sigma] \vdash \lambda z_1 \ldots z_n.x(\mathsf{Id}_0 y)(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n):\rho.$$

Using all the rst three points of Lemma 4.1 we get $\vdash \mathsf{Id}_0:\mathcal{C}'[\alpha\cap\sigma] \to \mathcal{C}'[\sigma]$: then by induction there is Id_0' such that $\mathsf{Id}_0 \longrightarrow_\eta \mathsf{Id}_0'$ and $\vdash \mathsf{Id}_0':\mathcal{C}'[\sigma] \to \mathcal{C}'[\alpha\cap\sigma]$ (notice that the hole occurrence is negative in $\mathcal{C}'[\ ]$). We can then choose Id' as the β-normal form of $\lambda xyz_1 \ldots z_n.x(\mathsf{Id}_0' y)(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n)$.

If $\mathcal{C}[\ ] = \rho \to \mathcal{C}'[\ ]$, then by Lemma 4.1(2)

$$x:\rho \to \mathcal{C}'[\sigma], y:\rho \vdash \lambda z_1 \ldots z_n.x(\mathsf{Id}_0 y)(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n):\mathcal{C}'[\alpha \cap \sigma].$$

Using again all the rst three points of Lemma 4.1, along with the typing rules, we get $t:\mathcal{C}'[\sigma] \vdash \lambda z_1 \ldots z_n.t(\mathsf{Id}_1 z_1)\ldots(\mathsf{Id}_n z_n):\mathcal{C}'[\alpha \cap \sigma]$: then by induction there is Id''

such that $\lambda t z_1 \ldots z_n.t(\mathsf{Id}_1 z_1) \ldots (\mathsf{Id}_n z_n) \longrightarrow_\eta \mathsf{Id}''$ and $\vdash \mathsf{Id}'' : \mathcal{C}'[\alpha \cap \sigma] \to \mathcal{C}'[\sigma]$. We can then choose Id' as the β-normal form of $\lambda xy.(\mathsf{Id}'' xy)$.

If $\mathcal{C}[\ ] = \rho \cap \mathcal{C}'[\ ]$ and $\mathcal{C}'[\ ]$ is an arrow type, then by Lemma 4.8(1) either $\vdash \mathsf{Id} : \mathcal{C}'[\sigma] \to \mathcal{C}'[\alpha \cap \sigma]$ or $\rho = \beta \cap \rho'$ and $\vdash \mathsf{Id} : \beta \to \mathcal{C}'[\alpha \cap \sigma]$. In the first case, by induction there is Id'' such that $\mathsf{Id} \longrightarrow_\eta \mathsf{Id}''$ and $\vdash \mathsf{Id}'' : \mathcal{C}'[\alpha \cap \sigma] \to \mathcal{C}'[\sigma]$. By Lemma 4.5(1) $\vdash \mathsf{Id} : \rho \to \rho$, which implies $\vdash \mathsf{Id}'' : \rho \to \rho$, since $\mathsf{Id} \longrightarrow_\eta \mathsf{Id}''$ by Lemma 7.2. So we can choose $\mathsf{Id}' = \mathsf{Id}''$ by Lemma 4.5(2). The second case implies $\vdash \mathsf{Id} : \mathcal{C}[\varphi] \to \mathcal{C}[\alpha \cap \sigma]$ for an arbitrary φ, so it is impossible. $\square$

An example showing the necessity of the condition $\not\vdash \mathsf{Id} : \mathcal{C}[\varphi] \to \mathcal{C}[\alpha \cap \sigma]$ is given by $\mathcal{C}_0[\ ] = (([\ ] \to \tau) \to \rho \cap \rho') \cap ((\alpha \cap \sigma \to \tau) \to \rho) \cap (\psi \to \psi)$, for we have both $\vdash \lambda xy.xy : \mathcal{C}_0[\sigma] \to \mathcal{C}_0[\alpha \cap \sigma]$ and $\vdash \lambda xy.xy : \mathcal{C}_0[\varphi] \to \mathcal{C}_0[\alpha \cap \sigma]$, but there is no f.h.i. which inhabits $\mathcal{C}_0[\alpha \cap \sigma] \to \mathcal{C}_0[\sigma]$.

By Lemma 7.3 we can conclude with the following theorem which ensures the soundness of an improved formulation of the erasure reduction rule.

THEOREM 7.4. *An equivalent formulation of the erasure reduction rule is:*

$$\mathcal{C}[\alpha \cap \sigma] \rightsquigarrow \mathcal{C}[\sigma]$$

if

- *either there is* Id *such that* $\vdash \mathsf{Id} : \mathcal{C}[\sigma] \to \mathcal{C}[\alpha \cap \sigma]$ *and* $\not\vdash \mathsf{Id} : \mathcal{C}[\varphi] \to \mathcal{C}[\alpha \cap \sigma]$ *where* φ *does not occur in* $\mathcal{C}[\alpha \cap \sigma]$ *and the hole occurrence is positive;*
- *or there is* Id *such that* $\vdash \mathsf{Id} : \mathcal{C}[\alpha \cap \sigma] \to \mathcal{C}[\sigma]$ *and* $\not\vdash \mathsf{Id} : \mathcal{C}[\alpha \cap \varphi] \to \mathcal{C}[\sigma]$ *where* φ *does not occur in* $\mathcal{C}[\alpha \cap \sigma]$ *and the hole occurrence is negative.*

8. 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 I with $\times$ replaced by $\cap$) and by the order relation induced by the preorder reported in Section 7. 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 [Bono et al. 2008], [Liquori and Ronchi Della Rocca 2007]. It might therefore be interesting to explore the role of intersection type isomorphisms in such contexts.

A prototypal isomorphism checker, directly obtained by the permutation-tree de nition 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`.

Acknowledgments. We would like to thank the anonymous referees of CSL'08 submission and of the present submission for their detailed remarks and helpful comments. We are indebted to Loris D'Antoni and Daniele Rispoli for the Prolog implementation of our isomorphism checker.

REFERENCES

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

Bono, V., Venneri, B., and Bettini, L. 2008. A typed lambda calculus with intersection types. *Theoretical Computer Science 398,* 1-3, 95–113.

Bruce, K., Di Cosmo, R., and Longo, G. 1992. Provable isomorphisms of types. *Mathematical Structures in Computer Science 2,* 2, 231–247.

Bruce, K. and Longo, G. 1985. Provable isomorphisms and domain equations in models of typed languages. In *STOC'85*, R. Sedgewick, Ed. ACM Press, Providence, 263 – 272.

Coppo, M. and Dezani-Ciancaglini, M. 1980. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic 21,* 4, 685–693.

Dezani-Ciancaglini, M. 1976. Characterization of normal forms possessing an inverse in the $\lambda\beta\eta$-calculus. *Theoretical Computer Science 2,* 3, 323–337.

Dezani-Ciancaglini, M., Di Cosmo, R., Giovannetti, E., and Tatsuta, M. 2008. On isomorphisms of intersection types. In *CSL'08*, M. Kaminski and S. Martini, Eds. Lecture Notes in Computer Science, vol. 5213. Springer-Verlag, Berlin, 461–477.

Di Cosmo, R. 1995. Second order isomorphic types. A proof theoretic study on second order λ-calculus with surjective pairing and terminal object. *Information and Computation 119,* 2, 176–201.

Di Cosmo, R. 2005. A short survey of isomorphisms of types. *Mathematical Structures in Computer Science 15*, 825–838.

Fiore, M., Di Cosmo, R., and Balat, V. 2006. Remarks on isomorphisms in typed lambda calculi with empty and sum types. *Annals of Pure and Applied Logic 141,* 1–2, 35–50.

Laurent, O. 2005. Classical isomorphisms of types. *Mathematical Structures in Computer Science 15*, 969–1004.

Liquori, L. and Ronchi Della Rocca, S. 2007. Intersection types à la Church. *Information and Computation 205,* 9, 1371–1386.

Martin, C. F. 1972. Axiomatic bases for equational theories of natural numbers. *Notices of the American Mathematical Society 19,* 7, 778.

Ronchi Della Rocca, S. 1988. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science 59,* 1-2, 1–29.

Soloviev, S. V. 1983. The category of finite sets and cartesian closed categories. *Journal of Soviet Mathematics 22,* 3, 1387–1400. English translation of the original paper in russian published in Zapiski Nauchyn Seminarov LOMI, v.105, 1981.

Received October 2008; revised March 2009; accepted April 2009.