On Isomorphism of “Functional” Intersection and Union Types*

Mario Coppo Mariangiola Dezani-Ciancaglini Ines Margaria Maddalena Zacchi
Dipartimento di Informatica Università di Torino, corso Svizzera 185, 10149 Torino, Italy

Type isomorphism is useful for retrieving library components, since a function in a library can have a type different from, but isomorphic to, the one expected by the user. Moreover type isomorphism gives for free the coercion required to include the function in the user program with the right type. The present paper faces the problem of type isomorphism in a system with intersection and union types. In the presence of intersection and union, isomorphism is not a congruence and cannot be characterised in an equational way. A characterisation can still be given, quite complicated by the interference between functional and non functional types. This drawback is faced in the paper by interpreting each atomic type as the set of functions mapping any argument into the interpretation of the type itself. This choice has been suggested by the initial projection of Scott’s inverse limit λ-model. The main result of this paper is a condition assuring type isomorphism, based on an isomorphism preserving reduction.

1 Introduction

In a typed λ-calculus the notion of *type isomorphism* is a particularisation of the general notion of isomorphism in category theory, with the requirement that the morphisms proving the isomorphism are λ-definable. More specifically, two *types* σ and τ are *isomorphic* if there are two λ-terms M and N of types $\sigma \to \tau$ and $\tau \to \sigma$, respectively, such that $M \circ N$ is $\beta\eta$-equal to the identity at type τ and $N \circ M$ is $\beta\eta$-equal to the identity at type σ ($M \circ N$ is short for $\lambda x.M(Nx)$, where x is fresh).

The importance of type isomorphism has been highlighted by Di Cosmo [11], who noted that the equivalence relation on types induced by the notion of isomorphism allows one to abstract from inessential details in the representation of data in programming languages. To distinguish isomorphic types can entail useless drawbacks; for instance, if a library contains a function of type $\sigma \wedge \tau \to \rho$, a request on a function of type $\tau \wedge \sigma \to \rho$ will not have success. Note that types as keys are actually used in Hoogle [14], an Haskell API search engine which allows one to search many standard Haskell libraries by either function name, or by approximate type signature. Neil Mitchell [15] remarks that in this application a suitable notion of “closeness” of types is needed, and isomorphism represents one of the possible meanings of type closeness. Recently, Díaz-Caro and Dowek [13] pointed out that in typed lambda-calculus, in programming languages, and in proof theory, isomorphic types are often identified. For example, the definitionally equivalent types are identified in Martin-Löf’s type theory and in the Calculus of Constructions. For this reason [13] proposes a type system in which λ-terms getting a type have also all types isomorphic to it.

In the simply typed λ-calculus, the isomorphism has been characterised by Bruce and Longo [3] using the **swap** equation: $\sigma \to \tau \to \rho \approx \tau \to \sigma \to \rho$. In richer λ-calculi, obtained from the simply typed one by adding other type constructors (like product types [19, 2, 20]) or by allowing higher-order types

*This work was partially supported by EU Collaborative project ASCENS 257414, ICT COST Action IC1201 BETTY, MIUR PRIN Project CINA Prot. 2010LHT4KM and Torino University/Compagnia San Paolo Project SALT.

Jakob Rehof (Ed.): Intersection Types and Related Systems (ITRS)
EPTCS 177, 2015, pp. 53–64, doi:10.4204/EPTCS.177.5

(System F [3, 11]), the set of equations characterising isomorphic types is obtained in an incremental way. A survey of these results is given by Di Cosmo in [12].

As pointed out in [10], [7], this incremental approach does not work when intersection and union types are considered. The isomorphism is no longer a congruence and that prevents to give it a finitary axiomatisation. The lack of congruence can be shown considering, for instance, the types

$$\sigma = \varphi_1 \to \varphi_2 \to \varphi_3 \quad \text{and} \quad \tau = \varphi_2 \to \varphi_1 \to \varphi_3.$$

They are isomorphic (by argument swapping), while, in general, both their intersection and their union with another type, for instance $\rho = \varphi_4 \to \varphi_5 \to \varphi_6$, are not. The reason is that σ and τ are isomorphic by argument swapping, while ρ is isomorphic to itself by identity.

The standard models of intersection and union types map types to subsets of any domain that is a model of the untyped λ-calculus, with the conditions that the arrow is interpreted as the function space constructor and the intersection and union operators as the corresponding set-theoretic operators [1]. Oddly enough, type equality in the standard interpretation of intersection types in λ-models does not imply type isomorphism [10] and it is so also for union types. This fact is due to the interference between atomic types, without functional behaviour, and functional types. For example, $\sigma \vee \tau \to \rho$ and $\tau \vee \sigma \to \rho$ are equal in all standard models, and isomorphic. In fact, the term $\lambda xy.xy$ has both the types $(\sigma \vee \tau \to \rho) \to \tau \vee \sigma \to \rho$ and $(\tau \vee \sigma \to \rho) \to \sigma \vee \tau \to \rho$; note that these isomorphic types are both functional, and this fact is exploited in the deductions. On the contrary, the considered types are no longer isomorphic when put in intersection or in union with an atomic type φ, although their interpretations remain equal; indeed, there is no λ-term mapping $(\tau \vee \sigma \to \rho) \vee \varphi$ to $(\sigma \vee \tau \to \rho) \vee \varphi$, or vice-versa, since when a functional type is put in union (or in intersection) with an atomic type, the possibility of exploiting its functional shape is lost. Despite these problems, a characterisation of type isomorphism is given in [7], by defining an (effective) notion of type similarity which turns out to correspond to isomorphism.

The existence of non-isomorphic, but semantically equal, types reveals a weakness of the type assignment system considered in [7], due essentially to the fact that atomic types do not have a functional behaviour. This assumption is indeed questionable in the pure λ-calculus, where everything is a function. A type system for intersection types in which type isomorphism contains type equality has been proposed in [6] by assuming each atomic type equivalent to a functional one, in such a way that they can be freely interchanged in any deduction.

In the present paper, we extend the result of [6] considering also union types. This extension is not trivial owing to the rather odd nature of union types. For instance, as remarked in [1], in systems with intersection and union types, subject reduction does not hold in general.

Following [6], each atomic type is interpreted as the set of constant functions returning values belonging to the set itself. This is realised by assuming that any atomic type φ is equivalent to $\omega \to \varphi$ (where ω is the type interpreted as the whole domain). This choice is motivated by the definition of initial projections in Scott's D_∞ λ-model [18] and from the relations between inverse limit models and filter models [4]. In D_∞ each element of the initial domain D_0 is projected in a constant function which returns itself when applied to any argument. As proved in [4], D_∞ is isomorphic to a filter λ-model built from a set of atomic types which correspond to compact elements of the initial domain D_0. This model equates φ to $\omega \to \varphi$ by construction. In an applicative setting it is sensible to assume a semantics in which a constant value (say, an integer), when used as a function, returns itself, independently of its argument, validating the present functional interpretation of atomic types.

Summary Section 2 presents the type assignment system with its properties, notably Subject Reduction and Subject Expansion. Section 3 introduces the notion of isomorphism. Section 4 defines a set

$$(Ax)\quad x:\sigma \vdash x:\sigma \qquad (\cong)\quad \frac{\Gamma \vdash M:\sigma \quad \sigma \cong \tau}{\Gamma \vdash M:\tau}$$

$$(\to I)\quad \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x.M:\sigma \to \tau} \qquad (\to E)\quad \frac{\Gamma_1 \vdash M:\sigma \to \tau \quad \Gamma_2 \vdash N:\sigma}{\Gamma_1,\Gamma_2 \vdash MN:\tau}$$

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

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

$$(\vee E)\quad \frac{\Gamma_1, x:\sigma \wedge \zeta \vdash M:\rho \quad \Gamma_1, x:\tau \wedge \zeta \vdash M:\rho \quad \Gamma_2 \vdash N:(\sigma \vee \tau) \wedge \zeta}{\Gamma_1,\Gamma_2 \vdash M[N/x]:\rho}$$

Figure 1: Typing rules.

of isomorphism preserving normalisation rules for types. Section 5 gives a notion of similarity between types in normal form which assures isomorphism. Section 6 draws some possible further work.

2 Type Assignment System

Let $\mathtt{A}$ be a denumerable set of atomic types ranged over by φ,ψ, and ω an atom not in $\mathtt{A}$. The syntax of types is given by:

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

As usual, parentheses are omitted according to the precedence rule "$\wedge$ and $\vee$ over $\to$" and "$\to$" associates to the right. Arbitrary types are ranged over by σ,τ,ρ,ζ.

The following equivalence asserts the functional character of atomic types, by equating them to arrow types. It also states that ω is the top type, viewing intersection and union set-theoretically.

Definition 2.1 (Semantic type equivalence)**.** *The* semantic equivalence relation $\cong$ on types *is defined as the minimal congruence such that :*

$$\varphi \cong \omega \to \varphi \qquad \omega \cong \omega \to \omega \qquad \sigma \cong \sigma \wedge \omega \qquad \sigma \cong \omega \wedge \sigma \qquad \omega \cong \sigma \vee \omega \qquad \omega \cong \omega \vee \sigma.$$

The congruence allows one to state that $\sigma \cong \sigma'$ and $\tau \cong \tau'$ imply $\sigma \wedge \tau \cong \sigma' \wedge \tau'$ and $\sigma \vee \tau \cong \sigma' \vee \tau'$. Moreover $\sigma \to \tau \cong \sigma' \to \tau'$ if and only if $\sigma \cong \sigma'$ and $\tau \cong \tau'$. Note that no other equivalence is assumed between types, for instance $\sigma \wedge \tau$ is different from $\tau \wedge \sigma$ and $\sigma \vee \tau$ is different from $\tau \vee \sigma$.

In the type assignment system considered in this paper types are assigned only to linear λ-terms. A λ-term is *linear* if each free or bound variable occurs exactly once in it. This is justified by the observation that type isomorphisms are realised by particular linear λ-terms, called "finite hereditary permutators" (see Definitions 3.1 and 3.3). This is not restrictive since it is easy to prove that the full system, without linearity restriction [1], is conservative over the present one. Therefore the types that can be derived for the finite hereditary permutators are the same in the two systems, so the present study of type isomorphism holds for the full system too.

Figure 1 gives the typing rules. As usual, *environments* associate variables to types and contain at most one type for each variable. The environments are relevant, i.e. they contain only the used premises. The domain of the environment Γ is denoted by $dom(\Gamma)$. When writing Γ_1,Γ_2 one convenes that $dom(\Gamma_1) \cap dom(\Gamma_2) = \emptyset$. It is easy to verify that $\Gamma \vdash M:\sigma$ implies $dom(\Gamma) = FV(M)$, where $FV(M)$ denotes the set of free variables of M.

The following rules are admissible.

$$(L)\ \frac{x:\sigma \vdash x:\tau \quad \Gamma, x:\tau \vdash M:\rho}{\Gamma, x:\sigma \vdash M:\rho} \qquad (\omega)\ \frac{dom(\Gamma) = FV(M)}{\Gamma \vdash M:\omega}$$

$$(C)\ \frac{\Gamma_1, x:\sigma \vdash M:\tau \quad \Gamma_2 \vdash N:\sigma}{\Gamma_1, \Gamma_2 \vdash M[N/x]:\tau} \qquad (\vee I')\ \frac{\Gamma, x:\sigma \vdash M:\rho \quad \Gamma, x:\tau \vdash M:\rho}{\Gamma, x:\sigma \vee \tau \vdash M:\rho}$$

$$(\vee E')\ \frac{\Gamma_1, x:\sigma \vdash M:\rho \quad \Gamma_1, x:\tau \vdash M:\rho \quad \Gamma_2 \vdash N:\sigma \vee \tau}{\Gamma_1, \Gamma_2 \vdash M[N/x]:\rho}$$

Remark that, considering only linear terms, cut elimination (rule (C)) corresponds to standard β-reduction, while for arbitrary terms parallel reductions are needed; for details see [1]. Therefore one can state:

Theorem 2.2 (SR)**.** *If $\Gamma \vdash M:\sigma$ and $M \longrightarrow^*_\beta N$, then $\Gamma \vdash N:\sigma$.*

The Subject Reduction Theorem allows one to show some properties useful in the following proofs.

Corollary 2.3. *1. If $\Gamma \vdash \lambda x.M:\sigma \to \rho$ and $\Gamma \vdash \lambda x.M:\sigma \to \zeta$, then $\Gamma \vdash \lambda x.M:\sigma \to \rho \wedge \zeta$.*

2. If $\Gamma \vdash \lambda x.M:\sigma \to \tau$ and $\Gamma \vdash \lambda x.M:\rho \to \tau$, then $\Gamma \vdash \lambda x.M:\sigma \vee \rho \to \tau$.

3. If $\Gamma \vdash \lambda x.M:\sigma \to \rho$ and $\Gamma \vdash \lambda x.M:\tau \to \zeta$, then $\Gamma \vdash \lambda x.M:\sigma \wedge \tau \to \rho \wedge \zeta$ and $\Gamma \vdash \lambda x.M:\sigma \vee \tau \to \rho \vee \zeta$.

In the considered system types are not preserved by η-reduction, as proved by the simple example:

$$\vdash \lambda xy.xy:\varphi \to \psi \to \varphi, \text{ but} \nvdash \lambda x.x:\varphi \to \psi \to \varphi$$

On the contrary, subject expansion holds for both β and η-expansions.

Theorem 2.4 (Subject Expansion)**.** *If M is a linear λ-term and $M \longrightarrow^*_{\beta\eta} N$ and $\Gamma \vdash N:\sigma$, then $\Gamma \vdash M:\sigma$.*

Proof. For β-expansion it is enough to show: $\Gamma \vdash M[N/x]:\sigma$ implies $\Gamma \vdash (\lambda x.M)N:\sigma$. The proof is by induction on the derivation of $\Gamma \vdash M[N/x]:\sigma$. The only interesting case is when the last applied rule is

$$(\vee E)\ \frac{\Gamma_1, x:\rho \wedge \zeta \vdash M:\sigma \quad \Gamma_1, x:\tau \wedge \zeta \vdash M:\sigma \quad \Gamma_2 \vdash N:(\rho \vee \tau) \wedge \zeta}{\Gamma_1, \Gamma_2 \vdash M[N/x]:\sigma}$$

It is easy to derive $x:(\rho \vee \tau) \wedge \zeta \vdash x:(\rho \wedge \zeta) \vee (\tau \wedge \zeta)$. Rule $(\vee I')$ applied to the first two premises gives $\Gamma_1, x:(\rho \wedge \zeta) \vee (\tau \wedge \zeta) \vdash M:\sigma$. So rule (L) derives $\Gamma_1, x:(\rho \vee \tau) \wedge \zeta \vdash M:\sigma$, and rule $(\to I)$ derives $\Gamma_1 \vdash \lambda x.M:(\rho \vee \tau) \wedge \zeta \to \sigma$. Rule $(\to E)$ gives the conclusion.
For η-expansion the proof is by induction on types. The only interesting case is when $\sigma = \tau \vee \rho$. Using rule $(\vee E')$ and applying the induction hypothesis to the first two assumptions one gets:

$$\frac{y:\tau \vdash \lambda x.yx:\tau \quad y:\rho \vdash \lambda x.yx:\rho \quad \Gamma \vdash M:\tau \vee \rho}{\Gamma \vdash \lambda x.Mx:\tau \vee \rho}$$

□

3 Isomorphism

The study of the type isomorphism in λ-calculus is based on the characterisation of λ-term invertibility. A λ-term P is *invertible* if there exists a λ-term P^{-1} such that $P \circ P^{-1} =_{\beta\eta} P^{-1} \circ P =_{\beta\eta} = \lambda x.x$. The paper [9] completely characterises the invertible λ-terms in the type free $\lambda\beta\eta$-calculus: the invertible terms are all and only the *finite hereditary permutators*.

Definition 3.1 (Finite Hereditary Permutator)**.** *A* finite hereditary permutator *(FHP for short) is a λ-term of the form (modulo β-conversion)*

$$\lambda xy_1 \ldots y_n.x(P_1 y_{\pi(1)}) \ldots (P_n y_{\pi(n)}) \quad (n \geq 0)$$

where π is a permutation of $1, \ldots, n$, and $P_1, \ldots, P_n$ are FHPs.

Note that the identity is trivially an FHP (take $n = 0$). Another example of an FHP is

$$\lambda xy_1y_2.xy_2y_1 \,{}_{\beta}^{*}\!\!\longleftarrow \lambda xy_1y_2.x((\lambda z.z)y_2)((\lambda z.z)y_1),$$

which proves the swap equation. It is easy to show that FHPs are closed under composition.

Theorem 3.2. *A λ-term is invertible iff it is a finite hereditary permutator.*

This result, obtained in the framework of the untyped λ-calculus, has been the basis for studying type isomorphism in different type systems for the λ-calculus. Note that every FHP P has, modulo $\beta\eta$-conversion, a unique inverse P^{-1}. Even if in the type free λ-calculus FHPs are defined modulo $\beta\eta$-conversion [9], in this paper FHPs are considered only modulo β-conversion, because types are not invariant under η-reduction. Taking into account these properties, the definition of type isomorphism can be stated as follows:

Definition 3.3 (Type Isomorphism)**.** *Two types σ and τ are* isomorphic *($\sigma \approx \tau$) if there exists a pair $< P, P^{-1} >$ of FHPs, inverse of each other, such that $\vdash P : \sigma \to \tau$ and $\vdash P^{-1} : \tau \to \sigma$. The pair $< P, P^{-1} >$ proves the isomorphism.*

When $P = P^{-1}$ one can simply write “P proves the isomorphism”.

It is immediate to verify that type isomorphism is an equivalence relation.

It is useful to single out FHPs, which only use the identity permutation, and the induced isomorphisms.

Definition 3.4 (Finite Hereditary Identity)**.** *A* finite hereditary identity *(FHI) is a λ-term of the form (modulo β-conversion)*

$$\lambda xy_1 \dots y_n.x(\mathsf{Id}_1 y_1)\dots(\mathsf{Id}_n y_n) \quad (n \geq 0)$$

where $\mathsf{Id}_1, \dots, \mathsf{Id}_n$ are FHIs.

The β-normal forms of FHIs are obtained from the identity $\lambda x.x$ through a finite (possibly zero) number of η-expansions. Then by Theorem 2.4 $\vdash \mathsf{Id} : \sigma \to \sigma$ for all FHIs Id and all σ.

Definition 3.5 (Strong Type Isomorphism)**.** *Two types σ and τ are* strongly isomorphic *($\sigma \approx_{\mathsf{S}} \tau$) if their isomorphism is proved by an FHI.*

Notice that requiring the isomorphism be proved by a pair of FHIs (instead of a single FHI) gives an equivalent definition of strong isomorphism, since types are preserved by η-expansion (Theorem 2.4).

Isomorphism does not imply strong isomorphism, for example $\lambda xyz.xzy$ proves $\omega \to \varphi \to \varphi \approx \varphi \to \varphi$, but $\omega \to \varphi \to \varphi \not\approx_{\mathsf{S}} \varphi \to \varphi$. Moreover semantic type equivalence implies strong type isomorphism, i.e. $\sigma \cong \tau$ implies $\sigma \approx_{\mathsf{S}} \tau$, but the inverse does not hold, since $\lambda x.x$ proves $\sigma \vee \tau \approx_{\mathsf{S}} \tau \vee \sigma$, but $\sigma \vee \tau \not\cong \tau \vee \sigma$.

It is useful to consider some strong isomorphisms, which are directly related to set theoretic properties of intersection and union and to standard properties of functional types. Moreover, all these isomorphisms are provable equalities in the system $\mathbf{B}_{+}$ of relevant logic [17].

Lemma 3.6. *The following strong isomorphisms hold:*

idem.	$\sigma \wedge \sigma \approx_{\mathsf{S}} \sigma$, $\sigma \vee \sigma \approx_{\mathsf{S}} \sigma$	**comm**.	$\sigma \wedge \tau \approx_{\mathsf{S}} \tau \wedge \sigma$, $\sigma \vee \tau \approx_{\mathsf{S}} \tau \vee \sigma$
assoc.	$(\sigma \wedge \tau) \wedge \rho \approx_{\mathsf{S}} \sigma \wedge (\tau \wedge \rho)$, $(\sigma \vee \tau) \vee \rho \approx_{\mathsf{S}} \sigma \vee (\tau \vee \rho)$		
dist$\wedge\vee$.	$(\sigma \vee \tau) \wedge \rho \approx_{\mathsf{S}} (\sigma \wedge \rho) \vee (\tau \wedge \rho)$	**dist**$\vee\wedge$.	$(\sigma \wedge \tau) \vee \rho \approx_{\mathsf{S}} (\sigma \vee \rho) \wedge (\tau \vee \rho)$
dist$\to \wedge$.	$\sigma \to \tau \wedge \rho \approx_{\mathsf{S}} (\sigma \to \tau) \wedge (\sigma \to \rho)$	**dist**$\to \vee$.	$\sigma \vee \tau \to \rho \approx_{\mathsf{S}} (\sigma \to \rho) \wedge (\tau \to \rho)$

Proof. The identity $\lambda x.x$ proves all these isomorphisms except the last two, proved by the η-expansion of the identity $\lambda xy.xy$. □

As regards to type interpretations, if σ is included in τ, the intersection $\sigma \wedge \tau$ is set-theoretically equal to σ and the union $\sigma \vee \tau$ is set-theoretically equal to τ. So, it is handy to introduce a pre-order on types which formalises set-theoretic inclusion and which takes into account the meaning of the arrow type constructor and the semantic type equivalence given in Definition 2.1. This pre-order is dubbed normalisation pre-order being used in the next section to define normalisation rules (Definition 4.1).

Definition 3.7 (Normalisation pre-order on types)**.** *The* normalisation relation $\leq$ on types *is the minimal pre-order relation such that:*

$$\sigma \leq \omega \qquad \sigma \wedge \tau \leq \sigma \qquad \sigma \wedge \tau \leq \tau \qquad \sigma \leq \sigma \vee \tau \qquad \tau \leq \sigma \vee \tau$$

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

$$\varphi \leq \sigma \to \varphi \qquad \omega \leq \sigma \to \omega \qquad \sigma' \leq \sigma,\, \tau \leq \tau' \Rightarrow \sigma \to \tau \leq \sigma' \to \tau'$$

Notice that $\sigma \leq \omega$ agrees with $\sigma \wedge \omega \cong \sigma$. Moreover $\varphi \leq \sigma \to \varphi$ and $\omega \leq \sigma \to \omega$ are justified by $\varphi \cong \omega \to \varphi$, $\omega \cong \omega \to \omega$ and the contra-variance of $\leq$ for arrow types.

The soundness of the normalisation pre-order follows from the following lemma, which shows the expected isomorphisms. To prove this lemma it is useful to observe that for each FHI Id, different from the identity, one gets $\mathsf{Id} \,{}^{*}_{\beta}\!\longleftarrow \lambda xy.\mathsf{Id}_1(x(\mathsf{Id}_2 y))$ for some FHIs $\mathsf{Id}_1, \mathsf{Id}_2$. For example, for

$$\mathsf{Id} = \lambda x y_1 y_2 y_3.x(\lambda t.y_1 t)y_2(\lambda u_1 u_2.y_3 u_1 u_2)$$

one has $\mathsf{Id}_1 = \lambda x y_2 y_3.x y_2(\lambda u_1 u_2.y_3 u_1 u_2)$ and $\mathsf{Id}_2 = \lambda xt.xt$.

The following lemma proves the validity of two more strong isomorphisms:

erase. if $\sigma \leq \tau$ then $\sigma \wedge \tau \approx_{\mathsf{s}} \sigma$ and $\sigma \vee \tau \approx_{\mathsf{s}} \tau$

Lemma 3.8. *1. If $\sigma \leq \tau$, then there is an FHI Id such that $\vdash \mathsf{Id} : \sigma \to \tau$.*

2. If $\sigma \leq \tau$, then $\sigma \wedge \tau \approx_{\mathsf{s}} \sigma$ and $\sigma \vee \tau \approx_{\mathsf{s}} \tau$.

Proof. (1). The proof is by induction on the definition of $\leq$. Only interesting cases are considered.
In case $\sigma \leq \rho$ and $\rho \leq \tau$ imply $\sigma \leq \tau$, by the induction hypothesis there are FHIs Id_1, Id_2 such that $\vdash \mathsf{Id}_1 : \sigma \to \rho$ and $\vdash \mathsf{Id}_2 : \rho \to \tau$. This implies $\vdash \lambda x.\mathsf{Id}_2(\mathsf{Id}_1 x) : \sigma \to \tau$. It is easy to verify that $\lambda x.\mathsf{Id}_2(\mathsf{Id}_1 x)$ reduces to an FHI.
In case $\sigma \leq \tau$ and $\sigma \leq \rho$ imply $\sigma \leq \tau \wedge \rho$, by the induction hypothesis there are FHIs Id_1, Id_2 such that $\vdash \mathsf{Id}_1 : \sigma \to \tau$ and $\vdash \mathsf{Id}_2 : \sigma \to \rho$. By Subject Reduction (Theorem 2.2) $\vdash \mathsf{Id}'_1 : \sigma \to \tau$ and $\vdash \mathsf{Id}'_2 : \sigma \to \rho$, where Id'_1 and Id'_2 are the β-normal forms of Id_1 and Id_2, respectively. By Subject Expansion (Theorem 2.4) there is an FHI Id, η-expansion of both Id'_1 and Id'_2, such that $\vdash \mathsf{Id} : \sigma \to \tau$ and $\vdash \mathsf{Id} : \sigma \to \rho$; by Corollary 2.3(1) $\vdash \mathsf{Id} : \sigma \to \tau \wedge \rho$. For the case $\sigma \leq \tau$ and $\rho \leq \tau$ imply $\sigma \vee \rho \leq \tau$, the proof is similar.
In case $\varphi \leq \sigma \to \varphi$, one can derive $y : \sigma \vdash y : \omega$ by rule (ω), and $x : \varphi \vdash x : \omega \to \varphi$ by rule $(\cong)$. Then $\vdash \lambda xy.xy : \varphi \to \sigma \to \varphi$ holds by rules $(\to E)$ and $(\to I)$.
In case $\sigma' \leq \sigma$ and $\tau \leq \tau'$ imply $\sigma \to \tau \leq \sigma' \to \tau'$, by the induction hypothesis there are FHIs Id_1, Id_2 such that $\vdash \mathsf{Id}_2 : \sigma' \to \sigma$ and $\vdash \mathsf{Id}_1 : \tau \to \tau'$. This implies $\vdash \lambda xy.\mathsf{Id}_1(x(\mathsf{Id}_2 y)) : (\sigma \to \tau) \to \sigma' \to \tau'$.
(2). By point (1) there is an FHI Id such that $\vdash \mathsf{Id} : \sigma \to \tau$. Clearly $\vdash \mathsf{Id} : \sigma \to \sigma$. Corollary 2.3(1) gives $\vdash \mathsf{Id} : \sigma \to \sigma \wedge \tau$. Since, obviously, $\vdash \lambda x.x : \sigma \wedge \tau \to \sigma$, Theorem 2.4 assures that Id proves the strong isomorphism $\sigma \wedge \tau \approx_{\mathsf{s}} \sigma$. In a similar way one proves that there is an Id proving $\sigma \vee \tau \approx_{\mathsf{s}} \tau$.

□

Strong isomorphism is a congruence, as shown in the following lemma, where *type contexts* are defined as usual:

$$C[\,] \ ::= \ [\,] \mid C[\,] \to \sigma \mid \sigma \to C[\,] \mid \sigma \wedge C[\,] \mid C[\,] \wedge \sigma \mid \sigma \vee C[\,] \mid C[\,] \vee \sigma$$

Lemma 3.9. *If $\sigma \approx_{\mathsf{s}} \tau$, then $C[\sigma] \approx_{\mathsf{s}} C[\tau]$.*

Proof. The proof is by structural induction on type contexts. For the empty context it is trivial. For any other context $\mathcal{C}[\ ]$, an FHI $\mathsf{Id}_{\mathcal{C}[\ ]}$ that proves the isomorphism $\mathcal{C}[\sigma] \approx_{\mathsf{S}} \mathcal{C}[\tau]$ is given by:

$$\mathsf{Id}_{\mathcal{C}[\ ]\to\rho\ \beta}\longleftarrow \lambda xy.x(\mathsf{Id}_{\mathcal{C}[\ ]}y) \qquad \mathsf{Id}_{\rho\to\mathcal{C}[\ ]\ \beta}\longleftarrow \lambda xy.\mathsf{Id}_{\mathcal{C}[\ ]}(xy)$$

$$\mathsf{Id}_{\rho\wedge\mathcal{C}[\ ]} = \mathsf{Id}_{\mathcal{C}[\ ]\wedge\rho} = \mathsf{Id}_{\mathcal{C}[\ ]} \qquad \mathsf{Id}_{\rho\vee\mathcal{C}[\ ]} = \mathsf{Id}_{\mathcal{C}[\ ]\vee\rho} = \mathsf{Id}_{\mathcal{C}[\ ]}$$

□

Owing to this lemma, types can be considered modulo idempotence, commutativity and associativity.

4 Normalisation

To investigate type isomorphism, following a common approach [2, 12, 10, 5], a notion of *normal form* of types is introduced. *Normal type* is short for type in normal form. The notion of normal form is effective, as shown by Theorem 4.3.

Type normalisation rules are introduced together with the proof of their soundness.

Definition 4.1 (Type normalisation rules)**.** *1. The inner type normalisation rules are:*

$(\varphi \rightsquigarrow)\quad \omega \to \varphi \rightsquigarrow \varphi$ $\qquad$ $(\omega \rightsquigarrow)\quad \omega \le \sigma$ *and* $\sigma \ne \omega$ *imply* $\sigma \rightsquigarrow \omega$

$(\wedge \rightsquigarrow)\quad \sigma \to \tau\wedge\rho \rightsquigarrow (\sigma\to\tau)\wedge(\sigma\to\rho)$ $\qquad$ $(\to_\wedge\rightsquigarrow)\quad (\sigma\vee\tau)\wedge\rho\to\zeta \rightsquigarrow (\sigma\wedge\rho)\vee(\tau\wedge\rho)\to\zeta$

$(\vee \rightsquigarrow)\quad \sigma\vee\tau\to\rho \rightsquigarrow (\sigma\to\rho)\wedge(\tau\to\rho)$ $\qquad$ $(\to_\vee\rightsquigarrow)\quad \sigma\to(\tau\wedge\rho)\vee\zeta \rightsquigarrow \sigma\to(\tau\vee\zeta)\wedge(\rho\vee\zeta)$

$(\le\rightsquigarrow)\quad \sigma\le\tau$ *implies* $\sigma\wedge\tau\rightsquigarrow\sigma$ *and* $\sigma\vee\tau\rightsquigarrow\tau$

2. The top type normalisation rules are:

$(ctx \Rightarrow)\quad \sigma\rightsquigarrow\tau$ *implies* $\mathcal{C}[\sigma]\Longrightarrow\mathcal{C}[\tau]$ $\qquad$ $(\vee\wedge\Rightarrow)\quad (\sigma\wedge\tau)\vee\rho\Longrightarrow(\sigma\vee\rho)\wedge(\tau\vee\rho)$

The first two rules follow immediately from semantic type equivalence; moreover, since $\omega\le\sigma\to\omega$, an admissible rule is $\sigma\to\omega\rightsquigarrow\omega$. The following four rules correspond to the distribution isomorphisms. The last rule corresponds to the **erase** isomorphism. Note that in the inner rules $(\to_\wedge\rightsquigarrow)$ and $(\to_\vee\rightsquigarrow)$ the isomorphism **dist**$\wedge\vee$ is used only on the left of an arrow and the isomorphism **dist**$\vee\wedge$ is used on the right of an arrow, respectively. These rules generate normal forms for arrow types in which the type on the left is an intersection and the type on the right is a union. Moreover the top rule $(\vee\wedge\Rightarrow)$ allows one to define for types a "conjunctive" normal form.

For example:

$$(\varphi_1\to\varphi_2)\wedge\varphi_2\rightsquigarrow\varphi_2 \text{ which implies } ((\varphi_1\to\varphi_2)\wedge\varphi_2)\vee\varphi_3\Longrightarrow\varphi_2\vee\varphi_3$$

$$(\varphi_1\to\varphi_2)\vee\varphi_2\rightsquigarrow\varphi_1\to\varphi_2 \text{ which implies } ((\varphi_1\to\varphi_2)\vee\varphi_2)\wedge\varphi_3\Longrightarrow(\varphi_1\to\varphi_2)\wedge\varphi_3$$

Having two kinds of normalisation rules (inner and top) allows to apply only one of the isomorphisms **dist**$\wedge\vee$ and **dist**$\vee\wedge$ at each subtype of a type. This is crucial to assure termination of normalisation.

The present normalisation rules are much simpler than those in [7]. The functional behaviour of atomic types produces this simplification.

Theorem 4.2 (Soundness of the normalisation rules)**.** *1. If* $\sigma\rightsquigarrow\tau$*, then* $\sigma\approx_{\mathsf{S}}\tau$*.*

2. If $\sigma\Longrightarrow\tau$*, then* $\sigma\approx_{\mathsf{S}}\tau$*.*

Proof. (1). Rule $(\varphi\rightsquigarrow)$ is obtained by orienting the equivalence relation between types, so it is sound since equivalent types are isomorphic. Rule $(\omega\rightsquigarrow)$ is sound because, by Lemma 3.8(1), there is an FHI Id such that $\vdash \mathsf{Id}:\omega\to\sigma$, and obviously $\vdash \mathsf{Id}:\sigma\to\omega$. Rules $(\wedge\rightsquigarrow)$, $(\to_\wedge\rightsquigarrow)$, $(\vee\rightsquigarrow)$ and $(\to_\vee\rightsquigarrow)$ are sound by the strong isomorphisms of Lemma 3.6. Lemma 3.8(2) implies the soundness of rule $(\le\rightsquigarrow)$.
(2). The soundness of the rule (ctx $\Rightarrow$) is proved in Lemma 3.9. The strong isomorphism **dist**$\vee\wedge$ gives the soundness of rule $(\vee\wedge\Rightarrow)$.

□

For example $((\varphi_1 \to \varphi_2) \wedge \varphi_2) \vee \varphi_3 \Longrightarrow \varphi_2 \vee \varphi_3$, as shown before, and $\lambda xy.xy$ proves

$$((\varphi_1 \to \varphi_2) \wedge \varphi_2) \vee \varphi_3 \approx_{\mathrm{s}} \varphi_2 \vee \varphi_3.$$

The following theorem shows the existence and uniqueness of the normal forms, i.e. that the top normalisation rules are terminating and confluent.

Theorem 4.3 (Uniqueness of normal form)**.** *The top normalisation rules of Definition 4.1 are terminating and confluent.*

Proof. The *termination* follows from an easy adaptation of the recursive path ordering method [8]. The partial order on operators is defined by: $\to\ \succ\ \vee\ \succ\ \wedge$ for holes at top level or in the right-hand-sides of arrow types and $\to\ \succ\ \wedge\ \succ\ \vee$ for holes in the left-hand-sides of arrow types. Notice that the induced recursive path ordering $\succ^*$ has the subterm property. This solves the case of rules $(\varphi \rightsquigarrow)$, $(\omega \rightsquigarrow)$, $(\leq\rightsquigarrow)$. For rule $(\wedge \rightsquigarrow)$, since $\to\ \succ\ \wedge$, it is enough to observe that $\sigma \to \tau \wedge \rho \succ^* \sigma \to \tau$ and $\sigma \to \tau \wedge \rho \succ^* \sigma \to \rho$. For rules $(\to_\wedge\rightsquigarrow)$ and $(\vee\wedge \Rightarrow)$, since $\vee\ \succ\ \wedge$ for holes at top level or in the right-hand-sides of arrow types, it is enough to observe that $(\sigma \wedge \tau) \vee \rho \succ^* \sigma \vee \rho$ and $(\sigma \wedge \tau) \vee \rho \succ^* \tau \vee \rho$. The proof for the remaining rules are similar.

For *confluence*, thanks to the Newman Lemma [16], it is sufficient to prove the convergence of the critical pairs. For example, the types $\sigma \vee \tau \vee \rho \to \zeta$, and $\sigma \to \tau \wedge \rho \wedge \zeta$ give rise to critical pairs, as well as the following ones, when $\sigma \leq \tau$:

$$(\sigma \wedge \tau) \vee \rho, \qquad \rho \to (\sigma \wedge \tau) \vee \zeta, \qquad (\sigma \vee \tau) \wedge \rho \to \zeta, \qquad \rho \to (\sigma \vee \tau) \wedge \zeta, \qquad \rho \vee (\sigma \wedge \tau) \to \zeta.$$

Other examples of critical pairs are $(\omega \to \varphi) \wedge \sigma$ if $\omega \to \varphi \leq \sigma$, and $\sigma \vee \omega$ if $\omega \leq \sigma$. □

The normal form of a type σ, unique modulo commutativity and associativity, is denoted by $\sigma\!\downarrow$. The soundness of the normalisation rules (Theorem 4.2) implies that each type is strongly isomorphic to its normal form.

Corollary 4.4. $\sigma \approx_{\mathrm{s}} \sigma\!\downarrow$.

As expected, semantic equivalent types have the same normal form. Clearly the inverse is false, since, for example, $(\sigma \to \tau \wedge \rho)\!\downarrow = (\sigma \to \tau) \wedge (\sigma \to \rho)$, but $\sigma \to \tau \wedge \rho \not\cong (\sigma \to \tau) \wedge (\sigma \to \rho)$.

Lemma 4.5. *If $\sigma \cong \tau$, then $\sigma\!\downarrow = \tau\!\downarrow$.*

Proof. The proof is by cases on Definition 2.1. For the equivalences $\varphi \cong \omega \to \varphi$ and $\omega \cong \omega \to \omega$, rules $(\varphi \rightsquigarrow)$ and $(\omega \rightsquigarrow)$ give $(\omega \to \varphi)\!\downarrow = \varphi$ and $(\omega \to \omega)\!\downarrow = \omega$, respectively. For the equivalences $\sigma \cong \omega \wedge \sigma$, $\sigma \cong \sigma \wedge \omega$, $\omega \cong \omega \vee \sigma$ and $\omega \cong \sigma \vee \omega$, rule $(\leq\rightsquigarrow)$, with $\sigma \leq \omega$, gives $(\omega \wedge \sigma)\!\downarrow = (\sigma \wedge \omega)\!\downarrow = \sigma$ and $(\omega \vee \sigma)\!\downarrow = (\sigma \vee \omega)\!\downarrow = \omega$. The congruence follows from rule (ctx $\Rightarrow$). □

5 Similarity as Isomorphism

This section shows the main result of the paper, i.e. that two types with "similar" normal forms (Definition 5.1) are isomorphic. The basic aim of the similarity relation is that of formalising isomorphism determined by argument permutations (as in the swap equation). This relation has to take into account the fact that, for two types to be isomorphic, it is not sufficient that they coincide modulo permutation of types in the arrow sequences, as in the case of cartesian products. Indeed the same permutation must be applicable to all types in the corresponding intersections and unions. The key notion of similarity exactly expresses such a condition.

To define similarity, it is useful to distinguish between different kinds of types. So in the following:

- α,β range over atomic and normal arrow types, i.e. $\alpha ::= \omega \mid \varphi \mid \xi \to \mu$;

- ξ,χ range over normal intersections of atomic and arrow types, i.e. $\xi ::= \alpha \mid (\xi \wedge \xi)\!\downarrow$;
- μ,ν range over normal unions of atomic and arrow types, i.e. $\mu ::= \alpha \mid (\mu \vee \mu)\!\downarrow$;
- η,θ range over normal types, i.e. $\eta ::= \mu \mid (\eta \wedge \eta)\!\downarrow$.

Definition 5.1 (Similarity)**.** *The* similarity *relation between two sequences of normal types $\langle \eta_1,\ldots,\eta_m\rangle$ and $\langle \theta_1,\ldots,\theta_m\rangle$, written $\langle \eta_1,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,\theta_m\rangle$, is the smallest equivalence relation such that:*

1. $\langle \eta_1,\ldots,\eta_m\rangle \sim \langle \eta_1,\ldots,\eta_m\rangle$
2. *if* $\langle \eta_1,\ldots,\eta_i,\eta_{i+1},\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,\theta_i,\theta_{i+1},\ldots,\theta_m\rangle$, *then*
$$\langle \eta_1,\ldots,(\eta_i \wedge \eta_{i+1})\!\downarrow,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,(\theta_i \wedge \theta_{i+1})\!\downarrow,\ldots,\theta_m\rangle \text{ and}$$
$$\langle \eta_1,\ldots,(\eta_i \vee \eta_{i+1})\!\downarrow,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,(\theta_i \vee \theta_{i+1})\!\downarrow,\ldots,\theta_m\rangle;$$
3. *if* $\langle \xi_i^{(1)},\ldots,\xi_i^{(m)}\rangle \sim \langle \chi_i^{(1)},\ldots,\chi_i^{(m)}\rangle$ *for* $1 \le i \le n$ *and* $\langle \mu_1,\ldots,\mu_m\rangle \sim \langle \nu_1,\ldots,\nu_m\rangle$, *then*
$$\langle (\xi_1^{(1)} \to \ldots \to \xi_n^{(1)} \to \mu_1)\!\downarrow,\ldots,(\xi_1^{(m)} \to \ldots \to \xi_n^{(m)} \to \mu_m)\!\downarrow\rangle \sim$$
$$\langle (\chi_{\pi(1)}^{(1)} \to \ldots \to \chi_{\pi(n)}^{(1)} \to \nu_1)\!\downarrow,\ldots,(\chi_{\pi(1)}^{(m)} \to \ldots \to \chi_{\pi(n)}^{(m)} \to \nu_m)\!\downarrow\rangle,$$
where π is a permutation of $1,\ldots,n$.

Similarity between normal types *is trivially defined as similarity between unary sequences: $\eta \sim \theta$ if $\langle\eta\rangle \sim \langle\theta\rangle$.*

For example, $\langle \varphi_1,\omega\rangle \sim \langle \varphi_1,\omega\rangle$, $\langle \omega,\varphi_2\rangle \sim \langle \omega,\varphi_2\rangle$, $\langle \varphi_3,\varphi_4\rangle \sim \langle \varphi_3,\varphi_4\rangle$ imply
$$\langle \varphi_1 \to \varphi_3, \omega \to \varphi_2 \to \varphi_4\rangle \sim \langle \omega \to \varphi_1 \to \varphi_3, \varphi_2 \to \varphi_4\rangle \text{ by (3) and then}$$
$$\langle (\varphi_1 \to \varphi_3) \vee (\omega \to \varphi_2 \to \varphi_4)\rangle \sim \langle (\omega \to \varphi_1 \to \varphi_3) \vee (\varphi_2 \to \varphi_4)\rangle \text{ by (2)} \quad .$$
This, together with $\langle\varphi_5\rangle \sim \langle\varphi_5\rangle$, $\langle\varphi_6\rangle \sim \langle\varphi_6\rangle$, $\langle\varphi_7\rangle \sim \langle\varphi_7\rangle$, gives
$$\langle \varphi_5 \to \varphi_6 \to \varphi_7 \to (\varphi_1 \to \varphi_3) \vee (\omega \to \varphi_2 \to \varphi_4)\rangle \sim \langle \varphi_7 \to \varphi_5 \to \varphi_6 \to (\omega \to \varphi_1 \to \varphi_3) \vee (\varphi_2 \to \varphi_4)\rangle$$
by (3).

The proof of the similarity soundness requires some ingenuity.

Theorem 5.2 (Soundness)**.** *If $\langle \eta_1,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,\theta_m\rangle$, then there is a pair of* FHPs *that proves $\eta_j \approx \theta_j$, for $1 \le j \le m$.*

Proof. By induction on the definition of $\sim$ (Definition 5.1).

(1). $\langle \eta_1,\ldots,\eta_m\rangle \sim \langle \eta_1,\ldots,\eta_m\rangle$. The identity proves the isomorphism.

(2). $\langle \eta_1,\ldots,(\eta_i \wedge \eta_{i+1})\!\downarrow,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,(\theta_i \wedge \theta_{i+1})\!\downarrow,\ldots,\theta_m\rangle$ since $\langle \eta_1,\ldots,\eta_i,\eta_{i+1},\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,\theta_i,\theta_{i+1},\ldots,\theta_m\rangle$. By the induction hypothesis there is a pair $< P, P^{-1} >$ that proves $\eta_j \approx \theta_j$, for $1 \le j \le m$. By Corollary 2.3(3), the same pair proves $\eta_i \wedge \eta_{i+1} \approx \theta_i \wedge \theta_{i+1}$. By Theorem 4.2 there are FHIs $\mathsf{Id}_1,\mathsf{Id}_2$ such that Id_1 proves $\eta_i \wedge \eta_{i+1} \approx (\eta_i \wedge \eta_{i+1})\!\downarrow$ and Id_2 proves $\theta_i \wedge \theta_{i+1} \approx (\theta_i \wedge \theta_{i+1})\!\downarrow$. Clearly $\vdash \mathsf{Id}_1 : \eta_j \to \eta_j$ and $\vdash \mathsf{Id}_2 : \theta_j \to \theta_j$ for $1 \le j \le m$. Then the pair $< \lambda x.\mathsf{Id}_2(P(\mathsf{Id}_1 x)), \lambda x.\mathsf{Id}_1(P^{-1}(\mathsf{Id}_2 x)) >$ proves the required isomorphisms. The proof for the case
$$\langle \eta_1,\ldots,(\eta_i \vee \eta_{i+1})\!\downarrow,\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,(\theta_i \vee \theta_{i+1})\!\downarrow,\ldots,\theta_m\rangle,$$
since $\langle \eta_1,\ldots,\eta_i,\eta_{i+1},\ldots,\eta_m\rangle \sim \langle \theta_1,\ldots,\theta_i,\theta_{i+1},\ldots,\theta_m\rangle$, is analogous.

(3).
$$\langle (\xi_1^{(1)} \to \ldots \to \xi_n^{(1)} \to \mu_1)\!\downarrow,\ldots,(\xi_1^{(m)} \to \ldots \to \xi_n^{(m)} \to \mu_m)\!\downarrow\rangle \sim$$
$$\langle (\chi_{\pi(1)}^{(1)} \to \ldots \to \chi_{\pi(n)}^{(1)} \to \nu_1)\!\downarrow,\ldots,(\chi_{\pi(1)}^{(m)} \to \ldots \to \chi_{\pi(n)}^{(m)} \to \nu_m)\!\downarrow\rangle$$
since $\langle \xi_i^{(1)},\ldots,\xi_i^{(m)}\rangle \sim \langle \chi_i^{(1)},\ldots,\chi_i^{(m)}\rangle$ for $1 \le i \le n$ and $\langle \mu_1,\ldots,\mu_m\rangle \sim \langle \nu_1,\ldots,\nu_m\rangle$. By the induction hypothesis, there are pairs $< P_i, P_i^{-1} >$ proving $\xi_i^{(j)} \approx \chi_i^{(j)}$ and a pair $< P_*, P_*^{-1} >$ proving $\mu_j \approx \nu_j$ for $1 \le i \le n$

and $1 \leq j \leq m$. Let

$$\begin{array}{lcl} P & = & \lambda x y_1 \ldots y_n.(P_*(x(P_1^{-1} y_{\pi^{-1}(1)}) \ldots (P_n^{-1} y_{\pi^{-1}(n)}))) \\ P^{-1} & = & \lambda x y_1 \ldots y_n.(P_*^{-1}(x(P_{\pi(1)} y_{\pi(1)}) \ldots (P_{\pi(n)} y_{\pi(n)}))) \end{array}$$

It is easy to verify that

$$\vdash P : (\xi_1^{(j)} \to \ldots \to \xi_n^{(j)} \to \mu_j) \to \chi_{\pi(1)}^{(j)} \to \ldots \to \chi_{\pi(n)}^{(j)} \to \nu_j$$
$$\vdash P^{-1} : (\chi_{\pi(1)}^{(j)} \to \ldots \to \chi_{\pi(n)}^{(j)} \to \nu_j) \to \xi_1^{(j)} \to \ldots \to \xi_n^{(j)} \to \mu_j$$

for $1 \leq j \leq m$. Notice that $(\xi_1 \to \ldots \to \xi_h \to \mu)\downarrow = \begin{cases} \xi_1 \to \ldots \to \xi_k \to \mu & \text{if } \xi_k \neq \omega \text{ and } \xi_{k+1} = \ldots = \xi_h = \omega \\ & \text{and } \mu \text{ is an atomic type,} \\ \xi_1 \to \ldots \to \xi_h \to \mu & \text{otherwise} \end{cases}$

since $\xi_1, \ldots, \xi_h$ are normal intersections of atomic and arrow types and μ is a normal union of atomic and arrow types. Then $\xi_1 \to \ldots \to \xi_h \to \mu \cong (\xi_1 \to \ldots \to \xi_h \to \mu)\downarrow$, and, by the typing rule ($\cong$):

$$\vdash P : (\xi_1^{(j)} \to \ldots \to \xi_n^{(j)} \to \mu_j)\downarrow \to (\chi_{\pi(1)}^{(j)} \to \ldots \to \chi_{\pi(n)}^{(j)} \to \nu_j)\downarrow$$
$$\vdash P^{-1} : (\chi_{\pi(1)}^{(j)} \to \ldots \to \chi_{\pi(n)}^{(j)} \to \nu_j)\downarrow \to (\xi_1^{(j)} \to \ldots \to \xi_n^{(j)} \to \mu_j)\downarrow$$

for $1 \leq j \leq m$. So $< P, P^{-1} >$ is the required pair.

□

An immediate implication of the Soundness Theorem and of Corollary 4.4 is that two types with similar normal forms are isomorphic.

Corollary 5.3. *If $\sigma\downarrow \sim \tau\downarrow$, then $\sigma \approx \tau$.*

For example the isomorphism of the types, shown similar after Definition 5.1, is proved by

$$< \lambda x y_1 y_2 y_3 y_4 y_5.x y_3 y_1 y_2 y_5 y_4, \lambda x y_1 y_2 y_3 y_4 y_5.x y_2 y_3 y_1 y_5 y_4 >.$$

6 Conclusion

This paper studies type isomorphism for a typed λ-calculus with intersection and union types, in which all types have a functional character. Atomic types become types of functions by assuming an equivalence relation that equates any atomic type φ to $\omega \to \varphi$. This equivalence has been introduced in [4] for constructing a filter model isomorphic to Scott's D_∞ and it is validated by the standard interpretation of types in this model. In the so obtained type system all types which are set-theoretically equal (using idempotence, commutativity, associativity and distributivity of intersection and union) are proved isomorphic by the identity.

Basic notions for the given development are those of type normalisation and similarity between normal types. Similarity provides a remarkable insight on isomorphism and we conjecture that, indeed, it gives a complete characterisation of type isomorphism for the system considered in the paper. We leave the proof of this conjecture as future work.

Following Díaz-Caro and Dowek [13] we aim to extend the type assignment systems developed in [5] and [7], by equating all isomorphic types. This would lead to introduce equivalence rules on λ-terms, see [13]. Lastly we plan to study type isomorphism in other assignment system with intersection and union types as, for instance, the ones for the lazy λ-calculus.

Acknowledgements The authors gratefully thank the referees and Alejandro Díaz-Caro for their numerous constructive remarks.

References

[1] Franco Barbanera, Mariangiola Dezani-Ciancaglini & Ugo de'Liguoro (1995): *Intersection and Union Types: Syntax and Semantics*. *Information and Computation* 119, pp. 202–230, doi:10.1006/inco.1995.1086.

[2] Kim Bruce, Roberto Di Cosmo & Giuseppe Longo (1992): *Provable Isomorphisms of Types*. *Mathematical Structures in Computer Science* 2(2), pp. 231–247, doi:10.1017/S0960129500001444.

[3] Kim Bruce & Giuseppe Longo (1985): *Provable Isomorphisms and Domain Equations in Models of Typed Languages*. In R. Sedgewick, editor: *STOC'85*, ACM Press, pp. 263 – 272, doi:10.1145/22145.22175.

[4] Mario Coppo, Mariangiola Dezani-Ciancaglini, Furio Honsell & Giuseppe Longo (1984): *Extended Type Structures and Filter Lambda Models*. In G. Lolli, G. Longo & A. Marcja, editors: *LC'82*, North-Holland, pp. 241–262.

[5] Mario Coppo, Mariangiola Dezani-Ciancaglini, Ines Margaria & Maddalena Zacchi (2013): *Towards Isomorphism of Intersection and Union Types*. In S. Graham-Lengrand & L. Paolini, editors: *ITRS'12*, *EPTCS* 121, pp. 58 – 80, doi:10.4204/EPTCS.121.5.

[6] Mario Coppo, Mariangiola Dezani-Ciancaglini, Ines Margaria & Maddalena Zacchi (2014): *Isomorphism of "Functional" Intersection Types*. In Ralph Matthes & Aleksy Schubert, editors: *Types'13*, 26, LIPIcs, pp. 129–149, doi:10.4230/LIPIcs.TYPES.2013.129.

[7] Mario Coppo, Mariangiola Dezani-Ciancaglini, Ines Margaria & Maddalena Zacchi (2014): *Isomorphism of Intersection and Union Types*. *Mathematical Structures in Computer Science*. To appear.

[8] Nachum Dershowitz (1982): *Orderings for Term-Rewriting Systems*. *Theoretical Computer Science* 17(3), pp. 279 – 301, doi:10.1016/0304-3975(82)90026-3.

[9] Mariangiola Dezani-Ciancaglini (1976): *Characterization of Normal Forms Possessing an Inverse in the $\lambda\beta\eta$-Calculus*. *Theoretical Computer Science* 2(3), pp. 323–337, doi:10.1016/0304-3975(76)90085-2.

[10] Mariangiola Dezani-Ciancaglini, Roberto Di Cosmo, Elio Giovannetti & Makoto Tatsuta (2010): *On Isomorphisms of Intersection Types*. *ACM TOCL* 11(4), pp. 1–22, doi:10.1145/1805950.1805955.

[11] Roberto Di Cosmo (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), pp. 176–201, doi:10.1006/inco.1995.1085.

[12] Roberto Di Cosmo (2005): *A Short Survey of Isomorphisms of Types*. *Mathematical Structures in Computer Science* 15, pp. 825–838, doi:10.1017/S0960129505004871.

[13] Alejandro Díaz-Caro & Gilles Dowek (2015): *Simply Typed Lambda-Calculus Modulo Type Isomorphisms*. *Theoretical Computer Science*. To appear.

[14] Neil Mitchell (2008): *Hoogle Overview*. *The Monad.Reader* 12, pp. 27–35.

[15] Neil Mitchell (2011): *Hoogle: Finding Functions from Types*. Available at http://community.haskell.org/~ndm/downloads/slides-hoogle_finding_functions_from_types-16_may_2011.pdf. Invited Presentation from TFP 2011.

[16] Maxwell H. A. Newman (1942): *On Theories with a Combinatorial Definition of "Equivalence"*. *Annals of Mathematics* 43(2), pp. 223–243, doi:10.2307/1968867.

[17] Richard Routley & Robert K. Meyer (1972): *The Semantics of Entailment III*. *Journal of Philosophical Logic* 1, pp. 192–208, doi:10.1007/BF00650498.

[18] Dana Scott (1972): *Continuous Lattices*. In F. W. Lawvere, editor: *Toposes, Algebraic Geometry, and Logic*, *LNM* 274, Springer-Verlag, pp. 97–136, doi:10.1007/BFb0073967.

[19] Sergei Soloviev (1983): *The Category of Finite Sets and Cartesian Closed Categories*. *Journal of Soviet Mathematics* 22(3), pp. 1387–1400, doi:10.1007/BF01084396. English translation of the original paper in Russian published in Zapiski Nauchnych Seminarov LOMI, v.105, 1981.

[20] Sergei Soloviev (1993): *A Complete Axiom System for Isomorphism of Types in Closed Categories*. In A. Voronkov, editor: *LPAR'93*, *LNCS* 698, Springer-Verlag, pp. 360–371, doi:10.1007/3-540-56944-8_71.