Automorphisms of Types in Certain Type Theories and Representation of Finite Groups

Sergei Soloviev

Automorphisms of types in certain type theories and representation of finite groups

SERGEI SOLOVIEV

IRIT, University of Toulouse-3, 118, route de Narbonne, 31062, Toulouse, France.
Email: soloviev@irit.fr, and *ITMO University, 49, Kronverkski prospekt, 197101, St. Petersburg, Russia (associated researcher)*

In memoriam Kosta Došen

The automorphism groups of types in several systems of type theory are studied. It is shown that in simply typed λ-calculus $\lambda^1\beta\eta$ and in its extension with surjective pairing and terminal object these groups correspond exactly to the groups of automorphisms of finite trees. In second-order λ-calculus and in Luo's framework (LF) with dependent products, any finite group may be represented.

1. Introduction

Saunders Mac Lane was deeply interested in studies of relations between (in his own words) 'algebra and neighbouring fields of algebra and geometry.' He wrote 'These studies have given me the lively impression that many of the ideas of algebra do indeed arise from these other fields, and that this origin highlights the sense in which the science of mathematics exemplifies the interdependence of its parts' (Mac Lane 1976). In this paper, certain algebraic constructions go towards logic, but their logical 'incarnation' illustrates the interdependence of the different parts of mathematics in a similar way.

Some combinatory algebras and associated semigroups and groups of untyped λ-terms were considered, e.g., in Barendregt (1984), ch. 21. Below, we study the groups of *typed* λ-terms. This gives the main results a more categorical flavour because the types may be seen as objects of certain categories. For example, the set of types isomorphic to a given type together with the set of isomorphisms between these types is a groupoid.

It is shown that arbitrary finite groups can be represented as groups of automorphisms of (a) second-order types (types of system F) and (b) dependent products in dependent type theories. (We considered the typed logical framework LF (Luo 1994), but the same construction will work for other systems with a dependent product like the calculus of constructions (Coquand and Huet 1988).)

To give a broader context to these results, we consider also the groups of automorphisms of simple types and show that they are exactly the groups of automorphisms of finite trees. We show that the class of the automorphism groups of types with pairing and terminal object reduces to the class of automorphism groups of simple types. This class

may be described also as class of groups that may be obtained from symmetric groups by cartesian product and wreath product (Babai 1995).

It is well known, that all finite groups can be represented as groups of automorphisms of certain mathematical structures, e.g., partially ordered sets, Cayley coloured diagrams, etc. From an algebraic point of view, this paper just adds a few more possible representations. However, as far as we know, this is a first study in literature where the automorphism groups of types are considered.

The interest of this study is that it shows the richness of algebraic structures based on the notion of isomorphism of types and opens new connections with other domains of mathematics and computer science. The notion of isomorphism of types plays an important role in the contemporary foundational research, e.g., in Homotopy Type Theory (HoTT) (HoTT 2013). Isomorphisms of second-order types are studied in connection with *ML*-style programming languages (Di Cosmo 1995). Dependent types and their isomorphisms are used in many popular proof assistants, such as Coq (Delahaye 1999).

Isomorphisms (and retractions) of types have also some very pragmatic applications that may go as far as cryptography. On a practical side, an interest of automorphisms may be that they do not change the types of data, and so may be used more 'discreetly.'

Technically, the results are not very difficult. There is a close connection with the study of isomorphisms of types in Di Cosmo (1995), but a principal difference is that some well-known methods such as normalization and characterization of invertible terms in λ-calculus needed to be 'reoriented': Di Cosmo used them to establish the existence of an isomorphism and to develop the algorithms of verification that the given types are isomorphic while we study the structure of the sets of invertible terms themselves in order to describe the groups of automorphisms. The higher order case required also the idea that Cayley coloured diagrams may be 'modelled' using the higher order types. In this case, an important role is played by α-equality based on the renaming of bound variables.

As a main source about invertible terms, we use the classical paper (Dezani 1976). We use also some fundamental results about normalization (Barendregt 1984; Hankin 1994; Di Cosmo 1995), otherwise we tried to make our proofs as self-contained as possible.

To address a larger audience of algebraists, logicians and topologists, an elementary presentation of the main formal systems from type theory and λ-calculus is included.

2. Formal systems

2.1. *Untyped lambda calculus*

The presentation in this subsection is mostly based on Hankin (1994).

Untyped λ-calculus is a formal system, i.e., symbolic language together with some rules for the manipulation of syntactic expressions (terms) of the language. From the point of view of computation, it is very powerful: It has the same computational power as Turing machines or any other equivalent formalism, such as partial recursive functions.

The syntax of untyped λ-calculus is very simple. The class of λ-terms consists of words constructed from the following alphabet: variables $x, y, z \ldots$; abstraction operator λ; parentheses (,). The class Λ of λ-terms is the least class such that

— if x is a variable, $x \in \Lambda$;
— if $M \in \Lambda$, then $(\lambda x.M) \in \Lambda$;
— if $M, N \in \Lambda$, then $(MN) \in \Lambda$.

In this formalism, everything is considered as an untyped function. The intuitive meaning of the three clauses above is that a function can be a variable or an application of a function to another function, or to be obtained from another function by abstraction (application of λ-operator to some variable x).

The symbol $\equiv$ denotes syntactic equality. Usual convention is to elide internal . and λ, assume that abstraction associates to the right, application to the left, and application precedes abstraction (that permits to omit some of the parentheses). For example,

$$\lambda x_1 x_2 x_3.MN_1N_2N_3 \equiv (\lambda x_1.(\lambda x_2.(\lambda x_3.(((MN_1)N_2)N_3)))).$$

Sometimes we will need to abbreviate even more. Given a list of indexes $i_1, \ldots, i_n$, instead of $A_{i_1} \ldots A_{i_n}$, we will write $A_{i_1 \div i_n}$. This abbreviation may be used with the convention above, for example, $\lambda x_{1 \div n}.MN_{1 \div n}$ will have the same meaning as $\lambda x_1 \ldots x_n.MN_1 \ldots N_n$.

In the term $\lambda x.M$, M is called the *scope* of λx. All occurrences of a variable x in some term that are not in the scope of any λx are called free. The set of all the free variables of M is denoted FV(M). All the free occurrences of x in M are bound by λx in $\lambda x.M$. The term without free variables is called closed.

Example 2.1. A well-known use of untyped λ-calculus is to define *combinators*, i.e., closed terms used as constants or to combine other terms:

— $\mathbf{I} \equiv \lambda x.x$ (identity combinator).
— $\mathbf{K} \equiv \lambda xy.x$ (first projection).
— $\mathbf{S} \equiv \lambda xyz.xz(yz)$ (generalized function application).
— $\mathbf{B} \equiv \lambda xyz.x(yz)$ (composition).
— The *fixed point combinator* $\mathbf{Y} \equiv \lambda f.((\lambda x.f(xx))(\lambda x.f(xx)))$ that may be used to define (partial) recursion.

Combinators will not be really used in this paper, this example only illustrates the traditional use of untyped λ-calculus as a universal model of computation.

The intended use of λ-terms as operators that are applied to other λ-terms requires an equality relation. Equality of untyped λ-terms is generated by three basic conversions (also called α-, β- and η-conversions).

Below, $[N/x]M$ denotes the substitution of N for all the free occurrences of x in M with renaming of bound variables. If several substitutions are used, it will be assumed that they associate to the right: $[N_1/x_1] \ldots [N_n/x_n]M$ will have the same meaning as $[N_1/x_1](\ldots([N_n/x_n]M)\ldots)$. Moreover, to shorten the formulas, we will use the notation with $\div$, that is, $[N_{1 \div n}/x_{1 \div n}]M$ will also mean $[N_1/x_1](\ldots([N_n/x_n]M)\ldots)$.

First, substitution is defined for the case when no renaming is needed.

— The variable convention and α-conversion.

Definition 2.2. ((Hankin 1994), Def. 2.7: change of bound variables, α-conversion.) M' is produced from M by a change of bound variable (without renaming) if $M \equiv C[\lambda x.N]$ and $M' \equiv C[\lambda y.[y/x]N]$, where y does not occur at all in N and $C[]$ is a context with one hole.

Definition 2.3. ((Hankin 1994), Def. 2.8: α-conversion and α-congruence.) M is α-congruent to N, written $M \equiv_\alpha N$, if N results from M by a series of changes of bound variable.

Two standard examples: $\lambda x.xy \equiv_\alpha \lambda z.zy$ but not $\lambda x.xy \equiv_\alpha \lambda y.yy$.

Convention 2.4. ((Hankin 1994), Def. 2.9: variable convention) If $M_1, \ldots, M_n$ occur in a certain context, then in these terms all bound variables are chosen to be different from free variables.

— Substitution (general case). As in Hankin (1994), to define substitution one needs

1. to identify α-congruent terms;
2. to consider the λ-terms as representatives of their equivalence classes;
3. to interpret the substitution $[N/x]M$ as an operation on equivalence classes, using representatives according to the variable convention.

With this strategy, substitution is defined as follows:

1. $[N/x]x \equiv N$.
2. $[N/x]y \equiv y$ if x is not y.
3. $[N/x](\lambda y.M) \equiv \lambda y.([N/x]M)$.
4. $[N/x](M_1 M_2) \equiv ([N/x]M_1)([N/x]M_2)$.

Because of variable convention, it is not possible that some free variable of N be 'captured' by λy.

— Now the equality $=$ in the untyped λ-calculus is defined as the smallest equivalence relation such that

- if $M \equiv_\alpha N$, then $M = N$;
- $(\lambda x.M)N = [N/x]M \ \ (\beta)$;
- $\lambda x.Mx = M \ \ (x \notin \mathrm{FV}(M)) \ \ (\eta)$;
- The relation $=$ is closed w.r.t. the rules:

$$\frac{M = M' \quad N = N'}{MN = M'N'} \qquad \frac{M = N}{\lambda x.M = \lambda x.N}.$$

Sometimes reflexivity, symmetry and transitivity are also included as rules but we do not need to do it explicitly since $=$ is defined as an equivalence relation. The equalities marked (β) and (η) are usually called β- and η-conversions (β-conversion corresponds to function application).

Example 2.5. For any terms M, N, R,

— $\mathbf{I}M = M$, $\mathbf{K}MN = M$, $\mathbf{B}MN = \lambda x.M(Nx)$, $\mathbf{S}MNR = MR(NR)$;

— For the *fixed point combinator* $\mathbf{Y}$ and any term F

$$\mathbf{Y}F = (\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))F = (\lambda x.F(xx))(\lambda x.F(xx)) =$$
$$= F((\lambda x.F(xx))(\lambda x.F(xx))) = F(\mathbf{Y}F)$$

(this explains why it is called the *fixed point combinator*);
— $\lambda x_{1\div n}.yx_{1\div n} = y$.

The first two items use only β-conversion (several times), the last uses only η.

The notion of equality in λ-calculus is closely connected with the notions of normal form and normalization.

Definition 2.6. (Cf. Hankin (1994), Def. 2.21.) If $M \in \Lambda$ and M has no subterms of the form $(\lambda x.R)S$, then M is a β-normal form.

By analogy, a $\beta\eta$-*nf* (sometimes we shall write merely *nf*) is a β-*nf* which also does not contain any subterms of the form $\lambda x.(Rx)$ with $x \notin \mathrm{FV}(R)$.

Normalization is the process based on *reductions* (oriented conversions) leading to a normal form. In the untyped λ-calculus, two reductions are considered: $(\lambda x.M)N \to [N/x]M$ (β) and $\lambda x.(Mx) \to M$ $(x \notin \mathrm{FV}(M))$ (η). The α-congruence is not used as a reduction but only to respect the variable convention. A reduction sequence cannot be extended only if it ends by an *nf*.

Remark 2.7. In difference from the untyped case, in typed λ-calculi, the so-called η-expansion is often considered instead of the η-reduction. It may be useful there because the number of expansions is bound by the 'depth' of types, but here it does not make much sense.

Definition 2.8. (Cf. Hankin (1994), Def. 3.21.)

— If $M \in \Lambda$, then M has a β-*nf* ($\beta\eta$-*nf*) if there exists an N such that $M = N$ and N is a β-*nf* ($\beta\eta$-*nf*).
— M weakly normalizes ($\mathcal{WN}(M)$) if there exists a finite reduction sequence starting with M and leading to some normal term.
— M strongly normalizes ($\mathcal{SN}(M)$) if all reduction sequences starting with M are finite.

In Λ, there are terms that are not $\mathcal{SN}$ but do have normal form.

For example, $(\lambda x.y)((\lambda x.xx)(\lambda x.xx))$ is $\mathcal{WN}$ but not $\mathcal{SN}$. The normal form is y (it is obtained if we do leftmost β-reduction), but β-reduction at the right may be iterated and give an infinite reduction sequence.

However, there are some obvious cases when $\mathcal{SN}$ is 'hereditary':

— $\mathcal{SN}(M) \Rightarrow \mathcal{SN}(Mx)$, $\quad \mathcal{SN}(M) \Rightarrow \mathcal{SN}(\lambda x.M)$;
— $\mathcal{SN}(N_1) \wedge \ldots \wedge \mathcal{SN}(N_m) \Rightarrow \mathcal{SN}(\lambda x_{1\div k}.zN_{1\div m})$.

In the untyped λ-calculus, not all the terms have normal forms. For example, the fixponts combinator $\mathbf{Y}$ does not have any normal form. Respectively, for some terms, the normalization process may not terminate. $M \to^* N$ will mean that there exists a reduction sequence from M to N. We write $\mathcal{WN}(M)$ if $M \to^* N$ for some normal N.

The $\beta\eta$-reduction in Λ does have the so called Church–Rosser property ($\mathcal{CR}$):

— If $M \to^* M_1$ and $M \to^* M_2$, then there exists N such that $M_1 \to^* N$ and $M_2 \to^* N$.

This implies the following Church–Rosser theorem:

Theorem 2.9. ((Barendregt 1984), Th. 3.3.9).

i. The $\beta\eta$-reduction has $\mathcal{CR}$ property.
ii. If $M_1 = M_2$, then there exists some term N such that $M_1 \to^* N$ and $M_2 \to^* N$.

This has two important consequences: (i) if M has a $\beta\eta$-normal form N, then $M \to^* N$, i.e., M is weakly normalizing; (ii) M does have at most one normal form: all reduction sequences that terminate, do terminate with the same nf.

Remark 2.10. The pure λ-calculus is a very powerful formalism. All computable functions are representable, but such representations use clever coding tricks. An alternative to this approach is to add constants with associated reduction rules (so called δ-rules) (Hankin 1994). Obviously, if some constants are added *without* any extra rules, it does not change the reduction properties. Still, as we shall see, it may be useful when some transformations of terms (such as *erasure* in typed λ-calculus) are considered.

2.2. *Finite hereditary permutations*

Now we consider the characterization of terms in the untyped λ-calculus that possess an inverse (Dezani 1976). In our presentation, we follow Dezani (1976) and Di Cosmo (1995). The theorem of Mariangiola Dezani–Ciancaglini formulated below plays a central role in the description of invertible terms not only in untyped λ-calculus, but also in various typed systems.

Definition 2.11. (Cf. Dezani (1976), p. 323.) Let M and N be normal terms. For M to be inverse of N means that both relations $\lambda x.M(Nx) \to^* \lambda x.x$ and $\lambda x.N(Mx) \to^* \lambda x.x$ are valid.

We take the following definition of the finite hereditary permutation (f.h.p.):

Definition 2.12. (Cf. Dezani (1976) and Di Cosmo (1995), def. 1.9.2.) An untyped λ-term M is an f.h.p. iff

— $M \equiv \lambda x.x$, or
— $M \equiv \lambda z.\lambda x_{\sigma(1)\div\sigma(n)}.zM_{1\div n}$ where σ is a permutation of the set $\{1,\ldots,n\}$ and $\lambda x_i.M_i$ is a f.h.p. for all $1 \leqslant i \leqslant n$.

The first variable of an f.h.p. after the λ-prefix will be called its head variable.

Remark 2.13. It follows immediately from this definition that f.h.p.'s are closed terms.

Example 2.14. The following terms are f.h.p.'s:

— $\lambda z.\lambda x_2 x_1.z x_1 x_2$.
— $\lambda z.\lambda x_2 x_1 x_3.z x_1 x_2(\lambda y_2 y_1.x_3 y_1 y_2)$.

As in Di Cosmo (1995), in our definition, the terms $M_{1\div n}$ themselves are not f.h.p.'s (an abstraction λx_i has to be applied). Dezani considered the terms $N_i x_{\sigma(i)}$ instead of M_i where N_i are f.h.p.'s, but then $N_i x_{\sigma(i)}$ are not β-normal ((Dezani 1976), p. 334). In difference from both Di Cosmo and Dezani, we apply permutation to the indexes in the prefix and not at the right under the application. When the erasures of typed λ-terms will be considered, it will help to reconstruct directly their type from the λ-prefix. In fact both definitions are related by α-conversion via $x_i \mapsto x_{\sigma^{-1}(i)}$ and thus are equivalent.

Let us notice that the f.h.p.'s are not necessarily normal, even in Di Cosmo's version. For example, if in the definition of an f.h.p. M_n is $x_{\sigma(n)}$, then an η-reduction is possible; if $M_{n-1} \equiv x_{\sigma(n-1)}$, then another η-reduction is possible afterwards, and similar η-reductions may be possible inside M_i.

One may notice that such η-reductions are the only reductions possible due to the definition of an f.h.p. Via these reductions, an f.h.p. always reduces to a unique normal form that is also an f.h.p.

The detailed technical proof may be already found in Dezani (1976); however, let us quote Di Cosmo (1995) who provides a brief explanation: 'One may easily show that the f.h.p.'s are typable terms... By the usual abuse of language we may then speak of typed f.h.p.'s. Recall now that all typed terms possess a (unique) normal form (see Barendregt (1984)).'

(Di Cosmo speaks here about the system $\lambda^1\beta\eta$ without pairing nor terminal object.)

We shall permit us one more abuse of language and call f.h.p.'s all terms that possess an nf, and this nf is an f.h.p.

The main result about invertible untyped λ-terms is given by the following theorem:

Theorem 2.15. (See Di Cosmo (1995), Theorem 1.9.1; cf. Dezani (1976), main theorem.) Let M be an untyped term that possesses a normal form. Then M is invertible iff it is an f.h.p.

2.3. *The calculus $\lambda^2\beta\eta\pi*$ and its subsystems*

The proof-theoretical presentation of $\lambda^2\beta\eta\pi*$ is closely related to Di Cosmo (1995), with some modifications that are explained below.

The class of types of $\lambda^2\beta\eta\pi*$ is constructed from the type variables $X, Y, Z \ldots$; constant $\mathbf{T}$; type constructors $\rightarrow, \times, \forall$ and parentheses (,). It is the least class Θ such that

— $\mathbf{T} \in \Theta$;
— if X is a type variable, $X \in \Theta$;
— if $A \in \Theta$ and X is a type variable, then $\forall X.A \in \Theta$;
— if $A, B \in \Theta$, then $(A \rightarrow B), (A \times B) \in \Theta$.

To omit some of the parentheses, it is assumed that (in order of priority) $\forall < \rightarrow < \times$, all operations associate to the right, and the convention that permits to elide internal $\forall$ and . is applied. For example,

$$\forall XY.X \rightarrow Y \rightarrow X \times X \times X \equiv (\forall X.(\forall Y.(X \rightarrow (Y \rightarrow (X \times (X \times X)))))).$$

Usually the types $A \to B$ are referred to as the arrow (or function) types, and $A \times B$ as the product types. The intended meaning of $\mathbf{T}$ is the terminal object in the categorical sense, and the $*$ below will stand for the unique term of type $\mathbf{T}$. The variables in types are bound by $\forall$. In the type $\forall X.A$, the scope of $\forall X$ is A. The α-equality and variable convention are extended to types. This permits to define the substitution of types into types in a way similar to the untyped λ-terms.

Before the notion of a term is introduced, it is useful to define the class of pre-terms, since such notions as the α-equality may be extended to this class. We assume that a countable set of term variables $x, y, z, \ldots$ is fixed.

The class of pre-terms is the smallest class Θ such that

— the types $A \in \Theta$, the term variables x and the constant $*$ are pre-terms;
— if M and N are pre-terms, then (MN) is a pre-term;
— if M and N are pre-terms, then $\langle M, N\rangle$ is a pre-term;
— if M is a pre-term, then p_1M, p_2M are pre-terms;
— if $A \in \Theta$ and M is a pre-term, then so is $\lambda x : A.M$;
— if X is a type variable and M is a pre-term, then $(\lambda X.M)$ is a pre-term.

In the pre-terms, there are two binders, λ and $\forall$ (it may be used inside types), but it is easily seen that the notion of α-equality, the variable convention and the definition of the substitution can be extended to pre-terms.

To define well-typed terms, we will introduce a deductive system closely related to the second-order propositional calculus. Below, $\Gamma, \Delta \ldots$ will denote the contexts of type declarations, i.e., the lists of typed term variables $x : A, y : B \ldots$ where each variable name $x, y, \ldots$ is used at most once. When there is no confusion with the notion of a context as a 'word with a hole' $C[]$, they will be called merely contexts.

The *typing judgements* are the expressions of the form $\Gamma \vdash M : A$ where Γ is a context, M is a pre-term and A is a type.

Well-formed terms (or merely terms) are pre-terms that are part of the typing judgements derivable in the deductive system below.

Remark 2.16. The term *environment* is used in Di Cosmo (1995), but in type theory (in particular, in the system LF considered below) 'context' is the usual term, and a unified terminology seems preferable. As in Di Cosmo (1995), the type variables $X, Y \ldots$ are not explicitely included in the context. For the calculus $\lambda^2\beta\eta\pi*$, it is a minor technical point, because (in difference from dependent type systems) the order of variables in the context is without importance. There exist other presentations where they are included and given type (or kind) Type, e.g., X : Type, Y : Type…. Another difference from Di Cosmo (1995) is that in our paper the bound term variables are typed. Di Cosmo used untyped bound variables in his $\lambda^2\beta\eta\pi*$ (but typed in its first-order part), and this is not very convenient in the presence of β^2 reduction because it requires substitution into the types of bound variables, cf. Longo et al. (1993).

Axioms:

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} \qquad\qquad \frac{}{\Gamma \vdash * : \mathbf{T}}$$

Rules:

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \to B}(\to -\text{intro}) \quad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}(\to -\text{elim})$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N\rangle : A \times B}(\times - \text{intro}) \quad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash p_1 M : A}, \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash p_2 M : B}(\times - \text{elim})$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \lambda X.M : \forall X.A}(\forall - \text{intro})* \quad \frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash MB : A[B/X]}(\forall - \text{elim})**$$

* For the type variable X not free in the type of any free term variable that occurs in the term M. This restriction is often formulated in such a way that X should not occur in the types of variables in Γ at all, but with α-equality our formulation is equivalent.

**For any type B.

Traditionally, in logic these rules are called $\to$ –intro, $\to$ –elim, $\times -$ intro, $\times -$ elim, $\forall -$ intro and $\forall -$ elim. They also represent λ-abstraction, application, pairing, projection, universal abstraction and universal application rules that are usually considered in λ-calculus.

To define the equality relation, the following basic equalities are considered:

$$(\beta)\ \ (\lambda x : A.M)N = M[N/x], \quad (\eta)\ \ \lambda x : A.(Mx) = x \ if\ x \notin \mathrm{FV}(M)$$

$$(\pi)\ \ p_i\langle M_1, M_2\rangle = M_i, \quad (\mathrm{SP})\ \ \langle p_1 M, p_2 M\rangle = M$$

$$(top)\ \ M = * \text{ if } M : \mathbf{T}$$

$$(\beta^2)\ \ (\lambda X.M)A = M[A/X], \quad (\eta^2)\ \ \lambda X.(MX) = M$$

The equality generated by $\beta, \eta, \pi, \mathrm{SP}, \mathrm{top}, \beta^2, \eta^2$ is denoted by $=_2$.

Remark 2.17. Di Cosmo does not say clearly whether $=_2$ is defined on pre-terms or well-typed terms. We do need it only for the case when M, N are well typed and have the same type in the same context. Di Cosmo carefully studied the normalization properties of $\lambda^2\beta\eta\pi*$ and its subsystems. For some of them, e.g., $\lambda^1\beta\eta$ and $\lambda^2\beta\eta$, the $\mathcal{SN}$ and $\mathcal{CR}$ were known long before. For the full $\lambda^2\beta\eta\pi*$ and $\lambda^2\beta\eta*$, he devised a special system reductions and had shown that it is $\mathcal{WN}$ and $\mathcal{CR}$. In all cases, the equality may be defined by the condition of coincidence of normal forms.

We shall need the following subsystems of $\lambda^2\beta\eta\pi*$ also used by Di Cosmo:

— The calculus $\lambda^2\beta\eta$ which is $\lambda^2\beta\eta\pi*$ without product types, pair, projections and $\mathbf{T}$. Equality for $\lambda^2\beta\eta$ is generated by $\beta, \eta, \beta^2, \eta^2$ and denoted $=_2$ as well (this 'abuse of notation' is justified by the properties of normalization, explained below).

— The calculus $\lambda^1\beta\eta\pi*$ which is $\lambda^2\beta\eta\pi*$ restricted to the first order or simple types. Equality for $\lambda^1\beta\eta\pi*$ is generated by $\beta, \eta, \pi, \mathrm{SP}, \mathrm{top}$ and denoted by $=_1$.

— The calculus $\lambda^1\beta\eta$ which is $\lambda^1\beta\eta\pi*$ without product types, pairing, projections and $\mathbf{T}$. Equality for $\lambda^1\beta\eta$ is generated by β, η and denoted $=_1$ (justified by normalization).

Normalization provides a decision procedure for $=_2$ and $=_1$. The normalization properties of the whole calculus $\lambda^2\beta\eta\pi*$ are rather complex. The detailed discussion may be found in Di Cosmo (1995), ch. 2.

Di Cosmo confronted the following problem. If we consider reductions corresponding to the equalities β, η, π, SP, top, β^2, η^2 oriented from left to right, then the resulting system of reductions is not $\mathcal{CR}$. There are critical pairs, for example,

$$M \overset{\eta}{\leftarrow} \lambda x : \mathbf{T}.(Mx) \overset{top}{\rightarrow} \lambda x : \mathbf{T}.(M*)$$

that do not reduce to any common term. To obtain a system of reductions that is $\mathcal{CR}$ for the whole calculus $\lambda^2\beta\eta\pi*$ and for $\lambda^1\beta\eta\pi*$, Di Cosmo adds the extra rewriting rules called gentop, $\mathrm{SP}_{\mathrm{top}}$ and η_{top}, all to deal with the constant $\mathbf{T}$ which is known to cause problems for $\mathcal{CR}$ (Di Cosmo (1995), p.69).

These supplementary rules do not change the equality relation. With these rules, he proves that $\lambda^1\beta\eta\pi*$ and $\lambda^2\beta\eta\pi*$ are effectively weakly normalizing (Di Cosmo (1995), Theorems 2.5.1 and 2.5.2). The $\mathcal{CR}$ property of the extended system of reductions guarantees the uniqueness of normal forms and the decidability of $=_2$ and $=_1$.

Di Cosmo also shows that many subsystems of $\lambda^1\beta\eta\pi*$ and $\lambda^2\beta\eta\pi*$ are $\mathcal{SN}$ and $\mathcal{CR}$ for ordinary system of reductions. As we already mentioned, the ordinary systems of reductions for $\lambda^2\beta\eta$ and $\lambda^1\beta\eta$ based on the equalities above are $\mathcal{SN}$ and $\mathcal{CR}$. For these subsystems, we can refer to the Theorems 3.2.3 and 3.1.19 of Di Cosmo (1995), respectively (though the proofs were known before). The reductions are taken from left to right, in particular, one takes η-reductions, not expansions. However, it will be useful to permit some limited η-expansions when we consider typed isomorphisms in connection with the f.h.p.s (see Sections 3–5).

To complete this section, let us say a few words about categorical properties of typed λ-calculi. It is well known that $\lambda^1\beta\eta\pi*$ may be seen as a free cartesian closed category (Asperti and Longo 1991; Lambek and Scott 1988). In this category, types play the role of objects, $\times$ plays the role of cartesian product, $\rightarrow$ of internal *hom*-functor, $\mathbf{T}$ of terminal object; morphisms from A to B are the equivalence classes (w.r.t. $=_1$) of closed λ-terms $\vdash M : A \rightarrow B$. In a very limited way, this fact will be used in Section 4. The reader will need only the very basic notions of category theory.

2.4. *Dependent type systems*

In this subsection, we consider Luo's typed logical framework (LF) (Luo 1994). In Section 5, we give a detailed proof of our main theorem about a representation of arbitrary finite groups by automorphisms of types in the second-order system $\lambda^2\beta\eta\pi*$. The detailed consideration of LF will make clear that the proof that we propose for $\lambda^2\beta\eta\pi*$ can be easily transferred to LF. The way how it may be obtained for many other known dependent type systems (like the calculus of constructions) and for type theories specified in LF will also be clear, though we do not consider these cases in this paper.

Because LF is mostly used to specify type theories, types in LF are called kinds (to distinguish them from types in the specified type theories).

In difference from the systems considered above, the well-formed kinds, contexts and terms of LF cannot be defined independently.

In LF, there are five forms of judgements:

— $\Gamma \vdash \mathbf{valid}$ (Γ is a valid context).

— $\Gamma \vdash K\ \mathbf{kind}$ (K is a kind in the context Γ).

— $\Gamma \vdash M : K$ (M is an object of the kind K).

— $\Gamma \vdash M = M' : K$ (M and M' are equal objects of the kind K).

— $\Gamma \vdash K = K'$ (K and K' are equal kinds in Γ).

First, we define simultaneously (by mutual induction) the classes of pre-terms, pre-kinds and pre-contexts. They are the smallest classes that satisfy the following definition. As before, a countable set of term variables is fixed, and there is also a constant Type.

— The constant Type and the expressions $\mathrm{El}(M)$ where M is a pre-term are pre-kinds. In LF, the constant Type represents a special kind whose elements are terms M that may be 'lifted' to kinds of the form $\mathrm{El}(M)$. Informally, $\mathrm{El}(M)$ may be read as 'elements of M.'

— If K, K' are pre-kinds, and x is a term variable then $(x : K)K'$ is a pre-kind. The variable x is bound in K' by $(x : K)$ (K' is the scope of $(x : K)$).

— The list of distinct term variables with pre-kinds $x_1 : K_1, \ldots, x_n : K_n$ is a pre-context.

— Each term variable x is a pre-term.

— If M, N are pre-terms, then (MN) is a pre-term.

— If M is a pre-term, K is a pre-kind and x a term variable, then $[x : K]M$ is pre-term. The variable x is bound in M by $[x : K]$ (M is its scope).

Remark 2.18. It follows immediately from this definition that for any pre-kind K, we have $K \equiv (x_{1\div n} : K_{1\div n})K_0$, where K_0 is either Type or has the form $\mathrm{El}(M)$.

The notion of α-equality, the variable convention and the definition of the substitution can be extended to pre-terms, pre-kinds and pre-contexts.

In the syntax of LF, $(x : K)K'$ denotes dependent product, and $[x : K]M$ denotes abstraction. In case when it is known that x is not free in K', we will write $K \to K'$ instead of $(x : K)K'$. In our joint works (Soloviev and Luo 2002; Luo et al. 2013) and in Luo (1994), the expression $(K)K'$ is used but in this paper the aim is to make the link with simply typed case more clear. As above, $\equiv$ is the syntactic identity up to the renaming of bound variables.

The well-formed terms, kinds and contexts are parts of the judgements that are derivable in the deductive system below. In fact, the theorem about the so-called pre-supposed judgements holds: It says that if a complex judgement is derivable, then the judgements that assure the well-formedness of its parts are derivable separately. For example, if $\Gamma \vdash M = N : K$ is derivable, then $\Gamma \vdash \mathbf{valid}, \Gamma \vdash K\ \mathbf{kind}, \Gamma \vdash M : K, \Gamma \vdash N : K$ are derivable. For details, see Luo et al. (2013) and Soloviev and Luo (2002).

Below, we give the principal inference rules of LF.

Contexts and assumptions

$$\frac{}{<>\vdash \mathbf{valid}} \qquad \frac{\Gamma \vdash K\ \mathbf{kind}\ \ x \notin \mathrm{FV}(\Gamma)}{\Gamma, x : K \vdash \mathbf{valid}} \qquad \frac{\Gamma, x : K, \Gamma' \vdash \mathbf{valid}}{\Gamma, x : K, \Gamma' \vdash x : K}$$

$$\frac{\Gamma_1, \Gamma_3 \vdash \mathbf{valid}\ \ \Gamma_1, \Gamma_2 \vdash \mathbf{valid}}{\Gamma_1, \Gamma_2, \Gamma_3 \vdash \mathbf{valid}}(wkn)$$
(where $\mathrm{FV}(\Gamma_2) \cap \mathrm{FV}(\Gamma_3) = \varnothing$).

General equality rules

$$\frac{\Gamma \vdash K\ \mathbf{kind}}{\Gamma \vdash K = K} \qquad \frac{\Gamma \vdash K = K'}{\Gamma \vdash K' = K} \qquad \frac{\Gamma \vdash K = K'\ \ \Gamma \vdash K' = K''}{\Gamma \vdash K = K''}$$

$$\frac{\Gamma \vdash M : K}{\Gamma \vdash M = M : K} \quad \frac{\Gamma \vdash M = M' : K}{\Gamma \vdash M' = M : K} \quad \frac{\Gamma \vdash M = M' : K\ \ \Gamma \vdash M' = M'' : K}{\Gamma \vdash M = M'' : K}$$

The kind type

$$\frac{\Gamma \vdash \mathbf{valid}}{\Gamma \vdash \mathrm{Type}\ \mathbf{kind}} \quad \frac{\Gamma \vdash A : \mathrm{Type}}{\Gamma \vdash \mathrm{El}(A)\ \mathbf{kind}} \quad \frac{\Gamma \vdash A = B : \mathrm{Type}}{\Gamma \vdash \mathrm{El}(A) = \mathrm{El}(B)}$$

Dependent product (kinds and terms)

$$\frac{\Gamma, x : K \vdash K'\ \mathbf{kind}}{\Gamma \vdash (x : K)K'\ \mathbf{kind}} \qquad \frac{\Gamma, x : K_1 \vdash K_1' = K_2'\ \ \Gamma \vdash K_1 = K_2}{\Gamma \vdash (x : K_1)K_1' = (x : K_2)K_2'}$$

$$\frac{\Gamma, x : K \vdash M : K'}{\Gamma \vdash [x : K]M : (x : K)K'} \qquad \frac{\Gamma, x : K_1 \vdash M_1 = M_2 : K\ \ \Gamma \vdash K_1 = K_2}{\Gamma \vdash [x : K_1]M_1 = [x : K_2]M_2 : (x : K_1)K}$$

$$\frac{\Gamma \vdash N : (x : K)K'\ \ \Gamma \vdash M : K}{\Gamma \vdash (NM) : [M/x]K'} \qquad \frac{\Gamma \vdash N = N' : (x : K)K'\ \ \Gamma \vdash M = M' : K}{\Gamma \vdash (NM) = (N'M') : [M/x]K'}$$

$$\frac{\Gamma, x : K \vdash M' : K'\ \ \Gamma \vdash M : K}{\Gamma \vdash ([x : K]M')M = [M/x]M' : [M/x]K'} \qquad \frac{\Gamma \vdash N : (x : K)K'\ \ x \notin \mathrm{FV}(\Gamma)}{\Gamma \vdash [x : K](Nx) = N : (x : K)K'}$$

Remark 2.19. Among the rules for dependent products, the first two treat the dependent product introduction, the next two the abstraction, the two after them the application and the last two represent the $\beta\eta$-rules for equality. We have omitted such structural rules as substitution and retyping (replacement of a kind in the context by an equal kind).

In Luo et al. (2013) and Soloviev and Luo (2002), the list of rules included in the definition of LF was larger and contained all these rules. In fact, there were seven substitution rules due to different forms of judgements in LF. For illustration, let us give just one:

$$\frac{\Gamma, x : K, \Gamma' \vdash k' : K'\ \ \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]k' : [k/x]K'}.$$

It was shown there that all structural rules (including *wkn*) can be eliminated and every deduction can be reduced to a canonical form without structural rules. Another (more common) way to put it is that all structural rules are admissible.

The LF described above is meant to be a 'core' system used to specify other type theories. The usual way to do that is via extensions of the language of LF with new constants and asserting appropriate computation rules.

Example 2.20. (Luo (1994), Section 9.2.1) The internal second-order logic (SOL) is introduced by declaring the following constants:

$$\vdash \mathrm{Prop} : \mathrm{Type} \quad \vdash \mathrm{Prf} : \mathrm{El}(\mathrm{Prop}) \to \mathrm{Type}$$
$$\vdash \forall : (A : \mathrm{Type})((\mathrm{El}(A) \to \mathrm{Prop}) \to \mathrm{Prop})$$
$$I_A : (A : \mathrm{Type})(P : \mathrm{El}(A) \to \mathrm{Prop})((x : \mathrm{El}(A))\mathrm{Prf}(P(x)) \to \mathrm{Prf}(\forall\, A\, P))$$
$$E_\forall : (A : \mathrm{Type})(P : \mathrm{El}(A) \to \mathrm{Prop})(R : \mathrm{Prf}(\forall\, A\, P) \to \mathrm{Prop})$$
$$((g : (x : \mathrm{El}(A))\mathrm{Prf}(P(x)))\mathrm{Prf}(R(I_A A P g))) \to (z : \mathrm{Prf}(\forall A P))\mathrm{Prf}(R(z))$$

and asserting the following computation rule:

$$E_\forall A P R f(I_A A P g) = f(g) : \mathrm{Prf}(R(I_A A P g))$$

In this approach, the constants declared in LF may be seen as an encoding or representation of the axioms and rules of another language and reflect their intended semantics. For example, the constant Prf represents the rule with one premise. The premise is a proposition (an element of the type Prop) and the result of its application is the type of its proofs. As to $\forall$, $\forall AP$ represents the universal proposition that corresponds to the family of propositions indexed by the elements of $\mathrm{El}(A)$ via $P : \mathrm{El}(A) \to \mathrm{Prop}$ (a particular case of such a P would be more familiar correspondence $x \in A \mapsto B(x)$). I_A and E_A represent certain forms of $\forall -$ intro and $\forall -$ elim. The role of computation rules is similar to that of conversions in λ-calculus or equalities between derivations, etc.

Example 2.21. Here is an example of a deduction in LF with constant Prop ('polymorphic identity' parameterized by $A : \mathrm{Type}$ applied to the constant Prop is equal to the identity of Prop):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathbf{valid}}}{\vdash \mathrm{Type}\ \mathbf{kind}}}{A : \mathrm{Type} \vdash \mathbf{valid}}}{A : \mathrm{Type} \vdash A : \mathrm{Type}}}{A : \mathrm{Type} \vdash \mathrm{El}(A)\ \mathbf{kind}}}{A : \mathrm{Type}, x : \mathrm{El}(A) \vdash \mathbf{valid}}}{A : \mathrm{Type}, x : \mathrm{El}(A) \vdash x : \mathrm{El}(A)}}{A : \mathrm{Type} \vdash [x : \mathrm{El}(A)]x : \mathrm{El}(A) \to \mathrm{El}(A)} \quad \dfrac{}{\vdash \mathrm{Prop} : \mathrm{Type}}}{\vdash ([A : \mathrm{Type}]([x : \mathrm{El}(A)]x))\mathrm{Prop} = [x : \mathrm{El}(\mathrm{Prop})]x : \mathrm{El}(\mathrm{Prop}) \to \mathrm{El}(\mathrm{Prop})}$$

The following lemmas are proved by induction on the length of canonical derivation (the formulations are taken from Soloviev (2015)).

Lemma 2.22. (Strengthening.) Let $\Gamma, \Gamma', \Gamma'' \vdash J$ be any judgement derivable in LF. If the variables from Γ' do not occur into Γ'' and J, then $\Gamma, \Gamma'' \vdash J$ is derivable.

Since in LF types (*kinds*) of variables may depend on terms (other variables), the variables cannot any more be freely permuted.

Let us consider the list of variables with kinds, $x_{1\div k} : K_{1\div k}$. Let $x_i \lhd x_j$ denotes that x_i occurs in the kind K_j of x_j. The same applies to prefixes like $(x_{1\div k} : K_{1\div k})K$ and $[x_{1\div k} : K_{1\div k}]K$.

Let $x_{1\div k} : K_{1\div k}$ be part of a valid context (respectively $(x_{1\div k} : K_{1\div k})K$, $[x_{1\div k} : K_{1\div k}]K$ be part of a derivable kind or term). In this case, the relation $\lhd$ generates a partial order on indexes $1, \ldots, k$ which we shall denote by $\lhd*$.

Lemma 2.23. Consider the judgements

$$\Gamma, x_{1\div n} : K_{1\div n}, \Gamma' \vdash \mathbf{valid},$$
$$\Gamma \vdash (x_{1\div n} : K_{1\div n})K_0 \,\mathbf{kind},$$
$$\Gamma \vdash [x_{1\div n} : K_{1\div n}]P \; : (x_{1\div n} : K_{1\div n})K_0$$

in LF. For any permutation σ that respects the order $\lhd*$,

$$\Gamma, x_{\sigma_1} : K_{\sigma(1)\div\sigma(n)} : K_{\sigma(1)\div\sigma(n)}, \Gamma' \vdash \mathbf{valid},$$
$$\Gamma \vdash (x_{\sigma(1)\div\sigma(n)} : K_{\sigma(1)\div\sigma(n)})K_0 \,\mathbf{kind},$$
$$\Gamma \vdash [x_{\sigma(1)\div\sigma(n)} : K_{\sigma(1)\div\sigma(n)}]P \; : (x_{\sigma(1)\div\sigma(n)} : K_{\sigma(1)\div\sigma(n)})K_0$$

are derivable in LF.

Besides standard equality rules, equality in LF is defined by the rules for dependent products, among them the rules that correspond to β- and η-conversions. This permits to define conversions in a more familiar way. The following proposition holds[†]:

Proposition 2.24.

1. Let J be an LF-judgement (of any of the five forms described above) and v an occurrence of an expression either of the form $([x : K]M)N$ or of the form $[x : K](Mx)$ with x not free in M. Let J' be obtained by replacement of v by the occurrence of $[N/x]M$ or M, respectively. If J is derivable in LF, then J' is derivable. (This is a form of 'subject reduction' property for β and η reduction.)
2. Let J be of the form $\Gamma \vdash K \,\mathbf{kind}$ or $\Gamma \vdash M : K$, respectively, and v belong to K (respectively, to M). Let K', respectively, M' be obtained from K (M) as above. If J is derivable, then $\Gamma \vdash K = K'$, respectively, $\Gamma \vdash M = M' : K$ is derivable.

A proof may be obtained by induction on canonical derivations that uses the properties of pre-supposed judgements mentioned above.

[†] It is a corrected version of an auxiliary proposition from Soloviev (2015) (Proposition 2.3). The formulation there contained a trivial error, and the proofs of main results do not require significant changes after the formulation is corrected.

Essentially, this proposition says that the $\beta\eta$-reducibility of the well-formed expressions in LF may be described in terms similar to other known systems of λ-calculus.

We shall use the fact that LF is $\mathcal{SN}$ and $\mathcal{CR}$ with respect to β- and η-reductions. As a reference, let us cite the thesis (Goguen 1994), and another more recent thesis (Marie-Magdeleine 2009). In these works, the so-called typed operational semantics was applied to prove $\mathcal{SN}$ and $\mathcal{CR}$ for a powerful extension of LF called UTT (Unified Type Theory). Now the $\mathcal{SN}$ and $\mathcal{CR}$ of the 'core' LF is an easy consequence.

Thus, the equality of the (well-typed) terms in LF may be defined by the condition that their normal forms coincide.

Remark 2.25. An LF-term in nf has in general the form $[x_{1\div n} : K_{1\div n}].z\,M_{1\div m}$, where $M_{1\div m}$ are normal and M_m is different from x_n (z may coincide with one of x).

3. Retractions, isomorphisms and automorphisms of types

The isomorphisms of simple types are studied for over 30 years, cf. Soloviev (1983), Bruce and Longo (1985) and Bruce et al. (1992). Later, the studies were extended, to include isomorphism of types in other systems of type theory: linear (Soloviev 1993; Dosen and Petric 1997; Balat and Di Cosmo 1999), with second-order types (Di Cosmo 1995), with empty and sum types (Fiore et al. 2006), with intersection and sum types (Coppo et al. 2017) and dependent types (Delahaye 1999; Soloviev 2015). Retractions were considered as well, together with isomorphisms or separately. See, for example, Bruce and Longo (1985), Stirling (2013) and Coppo et al. (2016) ‡.

3.1. *A general view*

Let us outline how to define the notion of isomorphism of types in a very abstract way.

Let us have some deductive system that has types (denoted $A, B, C \ldots$) and terms (denoted $M, N \ldots$). Many systems of type theory use *contexts*, i.e., the lists of type declarations for free variables, usually written as $\Gamma = x_{1\div n} : A_{1\div n}$. (In LF, it is said 'kinds' instead of 'types' because the word 'types' is reserved for terms of the kind Type.)

Let some context Γ be fixed. Assume that

— for any types A, B the set of 'morphisms' $[A, B]_\Gamma$ is defined;
— for any types A, B, C and terms $M \in [A, B]_\Gamma$ and $N \in [B, C]_\Gamma$, the term $N \circ M : [A, C]_\Gamma$ called composition is defined.
— For any type C, there exists a distinguished term $id_C \in [C, C]_\Gamma$ called identity.
— Some equivalence relation $\approx$ is defined on terms such that w.r.t. $\approx$ the composition is associative and the usual properties of identity are respected. (In fact these conditions may be slightly weakened: the composition needs to be defined only for terms from $[A, B]_\Gamma, [B, A]_\Gamma, [A, A]_\Gamma, [B, B]_\Gamma$.)

‡ The last work contains an extensive bibliography.

We do not require that the structure of category would be defined on the whole calculus (mostly not to go into complex matters related to contextual categories, etc.), though the conditions above define obviously some categorical structure for a fixed context Γ.

Based on these data, we can define the notions of *retraction*, *isomorphism* and *automorphism.*

Definition 3.1. Type A is called a retract of type B if there are terms $M \in [A, B]_\Gamma$ and $N : [B, A]_\Gamma$ such that $N \circ M \approx id_A$. The term M in this case is called coretraction and the term N retraction.

Definition 3.2. Types A and B are called isomorphic if there are terms $M : [A, B]_\Gamma$ and $N : [B, A]_\Gamma$ such that $N \circ M \approx id_A$ and $M \circ N \approx id_B$. The terms N and M themselves are called (mutually inverse) isomorphisms.

This definition implies obviously the uniqueness (up to $\approx$) of the inverse isomorphism to a given isomorphism M. The inverse is denoted usually by M^{-1}.

Definition 3.3. The isomorphisms $N \in [A, A]_\Gamma$ are called automorphisms of A.

— In ordinary simply typed λ-calculus with $\beta\eta$-equivalence as $\approx$, the set $[A, B]_\Gamma$ may be, for example, the set of all well-typed terms of type $A \to B$ in context Γ, with $id_C = \lambda x : C.x$ and $N \circ M = \lambda x : A.N(Mx)$.
— When retractions are studied, it is common to consider terms with contexts (Stirling 2013). No complete description of retractions represented by closed terms only (even in simply typed case) is known. An example of a retraction/coretraction pair that requires context is

$$\lambda x : A.\lambda y : B.x : A \to B \to A$$
$$u : B \vdash \lambda z : B \to A.zu : (B \to A) \to A$$

Due to β-reductions, u disappears from the composite term that is equivalent to id_A, but the context $\Gamma = u : B$ is necessary if we want the composition to be defined.
— In many cases, the sets $[A, B]$ that contain only closed terms (and do not depend on context) may be considered. For example, in simply typed λ-calculus (with or without pairing), all isomorphisms in normal form are closed terms. See, e.g., Di Cosmo (1995).
— In second-order λ-calculus, the context that contains free type variables occurring in A and B may be needed but it is enough to consider the terms that do not contain free term variables (cf. Di Cosmo (1995), ch. 1 and 5).
— In dependent types systems, one must admit arbitrary contexts, because A and B may contain free variables, and there is no neat separation between different sorts of variables (types may depend on terms) (Delahaye 1999; Soloviev 2015).
— In the case of LF, we define $[K, K']_\Gamma$ as the set of derivable judgements $\Gamma \vdash M : \overline{K} \to \overline{K'}$. Here, $\overline{K}, \overline{K'}$ are the $\beta\eta$-normal forms of K and K'. The composition and *id* are defined as usual. (There are two main reasons to consider normal forms and take $\to$ instead of dependent product. First, it is easy to show that if K' truly depends on $x : K$ (i.e., x occurs in $\overline{K'}$), then the term $M : (x : K)K'$ cannot be two-side invertible. Second,

if we consider $\beta\eta$-normal forms of K and K', they may not contain some free variables that are present in K and K': for example, take $y : \mathrm{Type} \vdash (x : \mathrm{El}(y))\mathrm{El}(([z : \mathrm{El}(y)]y)x)$. The normal form of $(x : \mathrm{El}(y))\mathrm{El}(([z : \mathrm{El}(y)]y)x)$ is $(x : \mathrm{El}(y))\mathrm{El}(y) =_{\mathrm{def}} \mathrm{El}(y) \to \mathrm{El}(y)$. In principle, the normal forms may be well-formed in a more narrow context Γ_0. Still, the isomorphism of K and K' may not hold in Γ_0 because the context Γ is necessary to prove the equality of kinds to their normal forms.)

— In some systems of type theory, there is no arrow type $A \to B$. The set $[A, B]_\Gamma$ in this case may be defined as the set of terms of type B in the context $\Gamma, y : A$. Composition of $s \in [A, B]_\Gamma$ and $t \in [B, C]_\Gamma$ can be defined using substitution: $M \circ N = [N/y]M$, and identity as $id_A \equiv y : A$ (cf. Soloviev (1983) and Di Cosmo (1995)).

If the types A and B are isomorphic (usually it will be clear with respect to which calculus and equivalence relation on terms), we shall write $A \sim B$.

3.2. *Erasure and invertibility*

A useful tool in the study of isomorphisms of types is *erasure*. We shall define erasure for $\lambda^1\beta\eta$, $\lambda^2\beta\eta$ and 'core' LF. In other cases, the erasure-based technique may require some δ-rules to be added to the definition of the equality in the untyped λ-calculus, but not in the above-mentioned cases. In case of $\lambda^2\beta\eta$, we shall consider the untyped λ-calculus extended by the constants $\to$ and $\forall$ but without δ-rules. The definition below also takes into account the differences in notation concerning standard definitions of $\lambda^1\beta\eta$, $\lambda^2\beta\eta$ and LF.

Definition 3.4. (Cf. Di Cosmo (1995), p.48.) Let us consider $\lambda^1\beta\eta$, $\lambda^2\beta\eta$ and 'core' LF. Let M be a typed λ-term, i.e., $\Gamma \vdash M : A$ in one of these systems.

The erasure $e(M)$ is defined as follows:

— Case of $\lambda^1\beta\eta$: $e(x) \equiv x$; $e(\lambda x : A.N) \equiv \lambda x.e(N)$; $e(M_1M_2) \equiv e(M_1)e(M_2)$. One may say merely that all type labels of variables are erased.

— Case of LF (we consider M being part of $\Gamma \vdash M : K$): $e(x) \equiv x$; $e([x : K]N) \equiv \lambda x.e(N)$; $e(M_1M_2) \equiv e(M_1)e(M_2)$.

— Case of $\lambda^2\beta\eta$. Here, we define first $e(B)$ for types, because types may occur not only as labels of variables. The untyped λ-calculus will contain now two sorts of variables: $x, y, z, \ldots$ and $X, Y, Z \ldots$ (inside the calculus, they are treated in exactly the same way, and introduced only to trace the origin of these variables) and is extended by two constants, $K_\forall$ and $K_\to$, without any supplementary δ-rules for equality.

  – Let B be a type. We define $e(X) \equiv X$, $e(B_1 \to B_2) \equiv K_\to(e(B_1)e(B_2))$, $e(\forall X.B_0) \equiv K_\forall(\lambda X.e(B_0))$ [§].

[§] The abstraction λ is introduced in $e(\forall X.A)$ to respect binding. This definition is inspired by the definition of erasure for second-order types in Bruce and Longo (1985). However, we modified the definition of $e(A \to B)$. Bruce and Longo used $e(A \to B) \equiv K_\to e(A)e(B)$ but with their definition erasure may create redexes, for example, $e(\forall X.(Y \to X)) \equiv K_\forall(\lambda X.K_\to Y X)$. With our definition, $e(A)$ is normal for any type A. In fact, when one is interested only in the relationship with f.h.p.'s, there are many other possibilities to define $e(A \to B)$ in such a way that no redexes are created, for example, as $K_\to(K_- A)(K_+ B)$.

– Let M be a term. Now, $e(x) \equiv x$, $e(\lambda x : A.N) \equiv \lambda x.e(N)$, $e(\lambda X.N) \equiv \lambda X.e(N)$, $e(NB) \equiv e(N)e(B)$, $e(M_1M_2) \equiv e(M_1)e(M_2)$.

Remark 3.5. The erasures of the axioms and rules for equality of $\lambda^1\beta\eta$ and of $\lambda^2\beta\eta$ are the axioms and rules of untyped λ-calculus, cf. for example, Di Cosmo (1995), Remark 1.9.4. The same is true for equality of terms in LF (when we consider erasure in LF, the equality of kinds is not concerned). For any type B in $\lambda^2\beta\eta$, $e(B)$ is in nf; λ under $\forall$ can not be used in a $\beta\eta$-reduction, the only case when the term $e(B)$ has a head variable is $B \equiv X$.

Lemma 3.6. Let $\Gamma \vdash M : K$ in $\lambda^1\beta\eta$, $\lambda^2\beta\eta$, or in LF. Then M and $e(M)$ have normal forms $nf(M)$ and $nf(e(M))$ (unique, as explained above) and

$$\begin{array}{ccc} M & \longrightarrow & nf(M) \\ \downarrow & & \downarrow \\ e(M) & \longrightarrow & nf(e(M)) \equiv e(nf(M)) \end{array}$$

where horizontal arrows denote arbitrary reduction sequences that end by the corresponding normal forms, and vertical arrows denote erasure.

Let $\Gamma \vdash N : K$ in the same calculus. If $M =_1 N$ (in $\lambda^1\beta\eta$), $M =_2 N$ (in $\lambda^2\beta\eta$), respectively, $\Gamma \vdash M = N : K$ in LF, then $e(M) = e(N)$ in untyped λ-calculus.

Proof. The existence of $nf(e(M))$ follows from the fact that any typable term in the untyped λ-calculus has an nf and this nf is unique. The rest is an immediate consequence of Remark 3.5.$\diamond$ □

Theorem 3.7. Let $\vdash M : A \to B$ and $\vdash N : B \to A$ be well-formed terms in $\lambda^1\beta\eta$. If M and N are mutually inverse isomorphisms, then $e(M)$ and $e(N)$ are mutually inverse f.h.p.'s. In particular, for any isomorphism M, the erasure $e(M)$ is an f.h.p.

Proof. It follows from $e(MN) \equiv (e(M)e(N))$ and our remark 3.5. Cf. also Di Cosmo (1995), Theorem 1.9.5.$\diamond$ □

To take into account type variables in $\lambda^2\beta\eta$, let us define the notion of the f.h.p. for $\lambda^2\beta\eta$ (2-f.h.p.) Our definition is a slightly modified version of definition 1.9.6 in Di Cosmo (1995). (Di Cosmo did not use type labels with variables inside second-order terms, but we think this does not agree with the notation used for other typed λ-calculi in this paper, and creates presentational difficulties when substitution is concerned.)

Definition 3.8. A term M of untyped λ-calculus with two sorts of variables and constants $\forall$ and $\to$ as in definition 3.4, is a 2-f.h.p. iff

— $M = \lambda x.x$, or
— $M = \lambda z.\lambda v_{\sigma(1)\div\sigma(n)}.z\,M_{1\div n}$ for some permutation $\sigma : n \to n$, and
  – if $\lambda v_i = \lambda x_i$, then $\lambda x_i.M_i$ is a 2-f.h.p.;
  – if $\lambda v_i = \lambda X_i$, then $M_i \equiv X_i$

  for all $1 \leqslant i \leqslant n$.

The next theorems may be proved in the same way as Theorem 3.7.

Theorem 3.9. (Cf. Di Cosmo (1995), Theorem 1.9.7, and Bruce and Longo (1985), Lemma 2.4 and Theorem 2.5.) Let $\vdash M : A \to B$ and $\vdash N : B \to A$ be well-formed terms in $\lambda^2\beta\eta$. If M and N are mutually inverse isomorphisms, then $e(M)$ and $e(N)$ are $\beta\eta$-equal to mutually inverse 2-f.h.p.'s. In particular, for any isomorphism M, the erasure $e(M)$ is $\beta\eta$-equal to a 2-f.h.p.

Theorem 3.10. If $\Gamma \vdash M : K \to K'$ and $\Gamma \vdash N : K' \to K$ are mutually inverse isomorphisms in LF, then their erasures are mutually inverse f.h.p.'s. In particular, for any isomorphism $\Gamma \vdash M : K \to K'$, the erasure $e(M)$ is an f.h.p.

Remark 3.11. The Theorems 3.7, 3.9 and 3.10 may be strengthened. In particular, the opposite implications may be proved (if $e(M)$ is an f.h.p., then M is an isomorphism etc.) A proof for $\lambda^2\beta\eta$ may be found in Bruce and Longo (1985) and Di Cosmo (1995); however, our definition of the $\lambda^2\beta\eta$ and 2-f.h.p. is slightly different, and by some reason, Di Cosmo does not prove this in case of $\lambda^1\beta\eta$, so we provide the proofs of our own in the next subsection. In fact we need a more detailed analysis of the structure of typed isomorphisms to be used in the proofs of the main results.

The case of LF is much more complex, and a detailed analysis of typed terms is required because too much information about *kinds* is lost after erasure. We give a proof outline of a similar strengthening (Theorem 3.25) with the purpose to use it in the future study of automorphisms in LF and to show the 'inner work' of the erasure technique in case of dependent type systems.

Let us note that Lemma 3.24 that precedes this theorem would be sufficient to prove the Theorem 5.6 in this paper (about automorphisms in LF), but it would be strange to leave out a more fundamental result.

3.3. *Structure of typed isomorphisms*

The fact that the erasure produces invertible terms when applied to typed isomorphisms permits us to establish that the isomorphism terms themselves must have similar structure. Let us consider some lemmas about more fine aspects of this structure. Afterwards, in 3.4, some examples will be given.

Let A be a type in $\lambda^1\beta\eta$. Since this calculus does not contain the constant T and its only type constructor is $\to$, A is either a type variable or may be written (without ambiguity) in the form $A_1 \to \ldots A_n \to p$ where p is a type variable and $A_1, \ldots, A_n$ are types of similar form.

The depth $d(A)$ of the formula A is defined by $d(p) = 0, d(A_1 \to \ldots A_n \to p) = \max_{1 \leqslant i \leqslant n} d(A_i) + 1$.

An equivalent definition would be to define $d(A \to B) = \max(d(A) + 1, d(B))$.

The definition of $d(A)$ may be updated for types in $\lambda^2\beta\eta$: $d(\forall X.A) =_{\text{def}} d(X \to A)$.

Next, lemma is closely related to some propositions and lemmas of Di Cosmo (1995) (Lemma 4.2.8, Propositions 4.3.1 and 4.3.3). However, we give a more direct proof based on the properties of f.h.p.'s.

Lemma 3.12. Let $\vdash M : A \to A'$ be an isomorphism in $\lambda^1\beta\eta$. Let $A \equiv A_1 \to A_2 \to \ldots A_n \to p$. Then there is a permutation $\sigma : n \to n$ such that

— $A' \equiv A'_{\sigma(1)} \to A'_{\sigma(2)} \to \ldots A'_{\sigma(n)} \to p$, where $A_i \sim A'_i (1 \leqslant i \leqslant n)$;
— $M =_1 \lambda z : A.\lambda x_{\sigma(1)\div\sigma(n)} : A'_{\sigma(1)\div\sigma(n)}.zM_{1\div n}$, where $M_{1\div n}$ are normal terms;
— the permutation σ and the terms $M_1, \ldots, M_n$ are uniquely determined by M;
— if $\vdash M' : A' \to A$ is the inverse isomorphism, then in a similar way $M' =_1 \lambda z' : A'.\lambda x'_{1\div n} : A_{1\div n}.z'M'_{\sigma(1)\div\sigma(n)}$;
— the terms $\lambda x_i : A_i.M'_i : A_i \to A'_i$ and $\lambda x_i : A'_i.M_i : A'_i \to A_i$ are mutually inverse isomorphisms $(1 \leqslant i \leqslant n)$.

Proof. We consider the terms first. □

We may assume that M is normal. Notice that its erasure $e(M)$ is then normal as well by Lemma 3.6. By Theorem 3.7, $e(M)$ is an f.h.p.

Thus, M has to be of the form $\lambda z : A.\lambda x_{\sigma_0(1)\div\sigma_0(m)} : A'_{\sigma_0(1)\div\sigma_0(m)}.zM_{1\div m}$ because $e(M)$ is $\lambda z.\lambda x_{\sigma_0(1)\div\sigma_0(m)}.ze(M_{1\div m})$.

The type of z is $A \equiv A_1 \to A_2 \to \ldots A_n \to p$. The types of $M_{1\div m}$ must match $A_1, A_2 \ldots$, thus $m \leqslant n$. The permutation $\sigma_0 : m \to m$ is uniquely determined by the condition that $\lambda x_i.e(M_i)$ is an f.h.p. More precisely, $x_{\sigma_0(i)}$ has to be the head variable of exactly one of the terms $e(M_1), \ldots, e(M_m)$ and this determines σ_0.

If $m = n$, the first three statements are immediate. Otherwise A may be written as

$$A_1 \to \ldots \to A_m \to (A_{m+1} \to \ldots \to A_n \to p).$$

Now the required structure is obtained by $n - m$ unique η-expansions and

$$M =_1 \lambda z : A.\lambda x_{\sigma(1)\div\sigma(m)} : A'_{\sigma(1)\div\sigma(m)}.\lambda x_{m+1\div n} : A_{m+1\div n}.zM_{1\div m}x_{m+1\div n}$$

where $\sigma(i) = \sigma_0(i), 1 \leqslant i \leqslant m$ and $\sigma(i) = i, m < i \leqslant n$, $M_{m+1} \equiv x_{m+1}, \ldots, M_n \equiv x_n$. In particular, the type of $zM_{1\div n}$ is p.

The same analysis applies to the inverse term $\vdash M' : A' \to A$ that should have a similar structure (with an obvious change of indexes). Notice, that up to this point, all statements about the structure of M, respectively, M', were derived from the assumption that the corresponding erasures are f.h.p.'s.

To prove the last statement, we consider the composition of M' and M. The result has to be equal to identity. We use the fact that, since the erasures are f.h.p.'s, the variable x_i occurs only as the head variable of M'_i, respectively, the variable x'_i occurs only as the head variable of M'_i $(1 \leqslant i \leqslant m)$. For $m < i \leqslant n$, these terms already are identities.

The statement about types is an immediate consequence of these facts. ⋄

Theorem 3.13. Let $\vdash M : A \to B$ and $\vdash N : B \to A$ in $\lambda^1\beta\eta$. If the erasure $e(M)$ is an f.h.p., then M is an isomorphism. Moreover, M can be reconstructed up to $=_1$ from the type $A \to B$ and the term $e(M)$. If $e(M)$ and $e(N)$ are mutually inverse f.h.p.'s., then M and N are mutually inverse isomorphisms.

Proof. We may assume that $A \equiv A_1 \to A_2 \to \ldots A_n \to p$ and $M, N, e(M), e(N)$ are normal. Proof of the first implication proceeds by induction on the size of $e(M)$.

Base case. If $e(M) \equiv \lambda x.x$, then $M \equiv \lambda x : A.x$.

Inductive step. As in the proof of lemma 3.12, if $e(M)$ is an f.h.p. different from identity, it has to be of the form $\lambda z.\lambda x_{\sigma_0(1)\div\sigma_0(m)}.ze(M_{1\div m})$, where $\lambda x_i.e(M_i)$ have to be f.h.p.'s as well ($\sigma_0 : m \to m, 1 \leqslant m \leqslant n$). It implies that $M \equiv \lambda z : A.\lambda x_{\sigma_0(1)\div\sigma_0(m)} : A'_{\sigma_0(1)\div\sigma_0(m)}.zM_{1\div m}$, the terms M_i are of types $A'_i \to A_i$, the variables x_i are their head variables, and since $e(\lambda x_i : A_i.M_i) \equiv \lambda x_i.e(M_i)$ are f.h.p.'s by I.H. the terms $\lambda x_i : A'_i.M_i$ are isomorphisms. Moreover, they can be reconstructed because we know the types $A'_i \to A_i$. They must have inverse isomorphisms of types $A_i \to A'_i$ that by Lemma 3.12 may be written in the form $\lambda x'_i : A_i.M'_i$. Direct computation shows that the terms M and $M' \equiv \lambda z : A.\lambda x_{1\div m} : A'_{1\div m}.zM'_{\sigma_0(1)\div\sigma_0(m)}$ are mutually inverse.

Now, if we consider $\vdash M : A \to B$ and $\vdash N : B \to A$, such that $e(M)$ and $e(N)$ are mutually inverse f.h.p.'s, then $e(MN) = e(M)e(N) = \lambda z.z$. Thus, $MN =_1 \lambda z : A.z$. Similar reasoning works for NM. Because the (two-side) inverse is uniquely determined, N is equal to M' constructed above. $\diamond$ □

One may notice that the second implication is proved very easily. In fact more important is the possibility to reconstruct an isomorphism from its erasure and type $A \to A'$, and it goes together with the first implication.

With some modification, similar results hold for $\lambda^2\beta\eta$.

The types in $\lambda^2\beta\eta$ have in general the following form:

$$(*)\ \forall X_{1\div i_1-1}(A_{i_1} \to \forall X_{i_1+1\div i_2-1}(A_{i_2} \to \ldots \forall X_{i_{n-1}+1\div i_n-1}(A_{i_n} \to \forall X_{i_n+1\div m}.X)\ldots)),$$

where $1 \leqslant i_1 \leqslant \ldots \leqslant i_n \leqslant m$. Each quantifier prefix may be empty (notationally, the usual convention that concerns indexes is applied). The types A_k ($1 \leqslant k \leqslant n$) have a similar structure. Below, we assume that there are no two bound variables with the same name and the names of all free variables are different from the names of bound variables. It means, in particular, that

(**) For any type variable Y and occurrence of a $\forall Y$ (in types) or of a λY (in terms) there is no occurrence of Y out of the scope of this $\forall Y$ (respectively, λY).

Lemma 3.14. Let $\vdash M : A \to A'$ be an isomorphism in $\lambda^2\beta\eta$. Let A be as above. There is a permutation $\sigma : m \to m$ such that

— $M =_2 \lambda z : A.\lambda v_{\sigma(1)\div\sigma(m)}.zP_{1\div m}$, where
  - $v_{i_k} \equiv x_{i_k} : A'_{i_k}$ ($1 \leqslant k \leqslant n$);
  - $v_j \equiv X_j$, when $1 \leqslant j \leqslant m, j \notin \{i_1, \ldots, i_n\}$;
  - $P_{i_1\div i_n}$ are normal terms of $\lambda^2\beta\eta$;
  - $P_j \equiv X_j$, when $1 \leqslant j \leqslant m, j \notin \{i_1, \ldots, i_n\}$;
  - the permutation σ and the expressions $P_1, \ldots, P_m$ are uniquely determined by M;

— if $j_1, \ldots, j_n$ are the elements of the set $\sigma^{-1}(\{i_1, \ldots, i_n\})$ ordered by $<$, then A' is

$$\forall X_{\sigma(1)\div\sigma(j_1-1)}(A'_{\sigma(j_1)} \to \ldots \forall X_{\sigma(j_{n-1}+1)\div\sigma(j_n-1)}(A_{\sigma(j_n)} \to \forall X_{\sigma(j_n+1)\div\sigma(m)}.X)\ldots);$$

— if $\vdash M' : A' \to A$ is the inverse isomorphism, then in a similar way

- $M' =_2 \lambda z' : A'.\lambda u'_{1\div m}.z' P'_{\sigma(1)\div\sigma(m)}$, where $u'_{i_1} \equiv x'_{i_1} : A_{i_1}, \ldots, u'_{i_n} \equiv x'_{i_n} : A_{i_n}$ and $u'_j \equiv X_j$, when $1 \leqslant j \leqslant m, j \notin \{i_1, \ldots, i_n\}$;
- $P'_{i_1\div i_n}$ are normal terms of $\lambda^2\beta\eta$ and $P'_j \equiv X_j, j \notin \{i_1, \ldots, i_n\}$;
- for any $k, 1 \leqslant k \leqslant n$, $\lambda x_{i_k} : A_{i_k}.P'_{i_k} : A_{i_k} \to A'_{i_k}$ and $\lambda x_{i_k} : A'_{i_k}.P_{i_k} : A'_i \to A_i$ are mutually inverse isomorphisms.

Proof. The term $e(M)$ is a 2-f.h.p by Theorem 3.9. We may assume that M and $e(M)$ are normal, and proceed as in Lemma 3.12 (including η-expansions). The main difference is due to the fact that now we have two sorts of variables, and need to take into account two possible cases, but it does not create any particular difficulties. The condition (**) will be respected if we respect the variable convention when α-conversion is applied. $\diamond$ □

This lemma is illustrated by Example 3.27 below.

Theorem 3.15. Let $\vdash M : A \to B$ and $\vdash N : B \to A$ in $\lambda^2\beta\eta$. If the erasure $e(M)$ is a 2-f.h.p., then M is an isomorphism. Moreover, M can be reconstructed up to $=_2$ from the type $A \to B$ and the term $e(M)$. If $e(M)$ and $e(N)$ are mutually inverse 2-f.h.p.'s., then M and N are mutually inverse isomorphisms.

Proof. We may assume that

$$A \equiv \forall X_{1\div i_1-1}(A_{i_1} \to \forall X_{i_1+1\div i_2-1}(A_{i_2} \to \ldots \forall X_{i_{n-1}+1\div i_n-1}(A_{i_n} \to \forall X_{i_n+1\div m}.X)\ldots)),$$

where $1 \leqslant i_1 \leqslant \ldots \leqslant i_n \leqslant m$ and $M, N, e(M), e(N)$ are normal.

Again, to prove the first implication, we proceed by induction on the size of $e(M)$.

Base case. If $e(M) \equiv \lambda x.x$, then $M \equiv \lambda x : A.x$. The case $e(M) \equiv \lambda X.X$ is excluded (it is not a 2-f.h.p.).

Inductive step. If $e(M)$ is an f.h.p. different from identity, it has to be of the form $\lambda z.\lambda v_{\sigma_0(1)\div\sigma_0(l)}.z e(P_{1\div l})$, where $\sigma_0 : l \to l, 1 \leqslant l \leqslant m$ and $\lambda v_i.e(P_i)$ is a 2-f.h.p. for $i \in \{i_1, \ldots, i_n\}$ and P_i is X_i otherwise.

It implies that $M \equiv \lambda z : A.\lambda x_{\sigma_0(1)\div\sigma_0(l)} : A'_{\sigma_0(1)\div\sigma_0(l)}.z P_{1\div l}$, for $i \in \{i_1, \ldots, i_n\}$ the terms P_i are of types $A'_i \to A_i$, the variables x_i are their head variables, and since $e(\lambda x_i : A_i.P_i) \equiv \lambda x_i.e(P_i)$ are 2-f.h.p.'s by I.H., the terms $\lambda x_i : A'_i.P_i$ are isomorphisms. Moreover, their types are known. So they can be reconstructed from $\lambda x_i.e(P_i)$ and type $A'_i \to A_i$.

They must have inverse isomorphisms of types $A_i \to A'_i$ that by Lemma 3.12 may be written in the form $\lambda x'_i : A_i.P'_i$. The remaining P_i and P'_i are the appropriate type variables not affected by erasure.

Direct computation shows that the terms $M' \equiv \lambda z : A.\lambda v'_{1\div l}.z P'_{\sigma_0(1)\div\sigma_0(l)}$ and M are mutually inverse.

Now, if we consider $\vdash M : A \to B$ and $\vdash N : B \to A$, such that $e(M)$ and $e(N)$ are mutually inverse 2-f.h.p.'s, then $e(MN) = e(M)e(N) = \lambda z.z$. Thus, $MN =_2 \lambda z : A.z$. Similar reasoning works for NM. Because the (two-side) inverse is uniquely determined, N is equal to M' constructed above. $\diamond$ □

The main subtlety in case of LF concerns the typing of M_j, while the somewhat artificial difference between term and type variables inherited from the standard formulation of $\lambda^2\beta\eta$ disappears.

The kinds in LF have in general the form $(x_{1\div n} : K_{1\div n})K_0$, where K_0 is either of the form $\mathrm{El}(A)$ or the constant Type.

Lemma 3.16. (Lemma 8 of Soloviev (2015).) Let $\Gamma \vdash \mathrm{El}(A)$ **kind** and $\Gamma \vdash \mathrm{El}(B)$ **kind**. Then, $\Gamma \vdash \mathrm{El}(A) \sim \mathrm{El}(B)$ iff $\Gamma \vdash \mathrm{El}(A) = \mathrm{El}(B)$. The isomorphism between $\mathrm{El}(A)$ and $\mathrm{El}(B)$ is identity in the sense that it is unique up to the equality in LF and is represented by the term $[x : \mathrm{El}(A).x]$ which is well typed in the context Γ:

$$\Gamma \vdash [x : \mathrm{El}(A)]x : (\mathrm{El}(A))\mathrm{El}(B).$$

Lemma 3.17. Let $\Gamma \vdash M : K \to K'$ be an isomorphism in LF. Let $(x_1 : K_1)\dots(x_n : K_n)K_0$ with K_0 either of the form $\mathrm{El}(A)$ or the constant Type. Then there is the unique permutation $\sigma : n \to n$ such that $K' \equiv K'_{\sigma(1)} \to K'_{\sigma(2)} \to \dots K'_{\sigma(n)} \to K'_0$ with K'_0 of the form $\mathrm{El}(A')$ or Type, respectively, and

$$M = [z : K].[x_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}].zM_{1\div n}$$

where the erasures $e([x_i : K'_i].M_i)$ are f.h.p.'s.

Proof. The proof proceeds along the same lines as the proof of Lemma 3.12. ⋄ □

Remark 3.18. There are some notable differences between Lemmas 3.17, 3.12 and 3.14:

— Since we did not establish anything yet about types and contexts of M_i, we do not assert that $[x_i : K'_i].M_i$ are isomorphisms.
— As above, to have a match between the number of premises K_i and the number of terms M_i some η-expansions may be used. In difference from previous cases, the kinds of variables added by η-expansions are not necessarily $K_{m+1}, \dots, K_n$ because $M_1, \dots, M_m$ may be substituted into kinds (see example 3.30 below).
— However, the LF-kinds may be used to model the types of $\lambda^2\beta\eta$. The dependent product over $(X : \mathrm{Type})$ may be used to represent $\forall X$, and $\to$ may be represented by $\to$ defined in LF ('dummy' product). It is possible to use this representation to obtain another proof of Lemma 3.14.

In Section 2.4, the relations $\lhd$ and $\lhd^*$ were introduced ($\lhd$ reflects the dependency between variables and $\lhd^*$ is its 'image' on natural numbers).

Lemma 3.19. (Lemma 19 of Soloviev (2015).) Let us consider $\Gamma \vdash M : K \to K'$, $\Gamma \vdash M' : K' \to K$ such that

— $e(M)$ and $e(M')$ are mutually inverse f.h.p.'s,
— $\Gamma \vdash M : K \to K'$, $\Gamma \vdash M' : K' \to K$ are derivable in LF,
— $M \equiv [z : (x_{1\div n} : K_{1\div n})K_0][x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}](zM_{1\div n})$ (M_i has x_i as its head variable),
— $M' \equiv [z' : (x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0][x_{1\div n} : K_{1\div n}](z'M'_{\sigma(1)\div\sigma(n)})$ (M'_i has x'_i as its head variable).

Then, $x_i \lhd x_j$ iff $x'_i \lhd x'_j$.

Corollary 3.20. The partial order $\lhd^*$ generated by $\lhd$ on $x_1, \dots, x_n$ coincides with $\lhd^*$ generated by $\lhd$ on $x'_1, \dots, x'_n$.

Lemma 3.21. (Lemma 21 of Soloviev (2015).) Let $M \equiv [x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}].zM_{1\div n}$ as above. Then x'_i, the head variable of M_i, does not occur in $M_j, j < i$. Similar result holds for M'.

Remark 3.22. This is stronger than a similar fact about occurrences into kinds mentioned before.

The next lemma permits to separate the 'permutation of the premises' and the isomorphisms between these premises.

Let $\Gamma \vdash M : K \to K'$, $\Gamma \vdash M' : K' \to K$, where $K \equiv (x_{1\div n} : K_{1\div n})K_0$, $K' \equiv (x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0$ and $e(M), e(M')$ be mutually inverse f.h.p.'s. We assume that the structure of M and M' is as established in the previous lemmas, i.e.

$$M \equiv [z : (x_{1\div n} : K_{1\div n})K_0][x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}](zM_{1\div n}),$$
$$M' \equiv [z' : (x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0][x_{1\div n} : K_{1\div n}](z'M'_{\sigma(1)\div\sigma(n)}).$$

Let $y_1, \ldots, y_n$ be fresh variables,

$$Q_1 \equiv K'_1,\ Q_2 \equiv [y_1/x'_1]K'_2, ..., Q_n \equiv [y_{n-1\div 1}/x'_{n-1\div 1}]K'_n, Q_0 \equiv [y_{n-1\div 1}/x'_{n-1\div 1}]K'_0.$$

Notice that

$$Q \equiv (y_{1\div n} : Q_{1\div n})Q_0 =_\alpha (x'_{1\div n} : K_{1\div n})K'_0,$$
$$(y_{\sigma(1)\div\sigma(n)} : Q_{\sigma(1)\div\sigma(n)})Q_0 =_\alpha (x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0 \equiv K'.$$

Let

$$P_\sigma \equiv [u : (y_{1\div n} : Q_{1\div n})Q_0][x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}](ux'_{1\div n})$$

and

$$P_\sigma^{-1} \equiv [v : (x_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0][y_{1\div n} : Q_{1\div n}](vy_{\sigma(1)\div\sigma(n)}).$$

Consider also

$$N \equiv [z : (x_{1\div n} : K_{1\div n})K_0][x'_{1\div n} : K'_{1\div n}](zM_{1\div n})$$

and

$$N' \equiv [z' : (x'_{1\div n} : K'_{1\div n})K'_0][x_{1\div n} : K_{1\div n}](z'M'_{1\div n}).$$

Next lemma is a more fine-grained formulation of Lemma 22 of Soloviev (2015).

Lemma 3.23. Under the assumptions above (in particular, that $\Gamma \vdash M : K \to K'$, $\Gamma \vdash M' : K' \to K$),

— $\Gamma \vdash P_\sigma : Q \to K'$, $\Gamma \vdash P_\sigma^{-1} : K' \to Q$, $\Gamma \vdash N : K \to Q$, $\Gamma \vdash N' : Q \to K$ are derivable in LF;
— P_σ and P_σ^{-1} are mutually inverse isomorphisms;
— $\Gamma \vdash M = P_\sigma \circ N : K \to K'$ and $\Gamma \vdash P_\sigma^{-1} \circ M = N : K \to Q$;
— $\Gamma \vdash M' \circ P_\sigma = N' : Q \to K$ and $\Gamma \vdash M' = N' \circ P_\sigma^{-1} : K' \to K$;
— $\Gamma \vdash M' \circ M = M' \circ (P_\sigma \circ N) = (M' \circ P_\sigma) \circ N = N' \circ N : K \to K$;
— $e(N)$ and $e(N')$ are mutually inverse f.h.p.'s iff $e(M)$ and $e(M')$ are;
— M and M' are mutually inverse isomorphisms iff N and N' are.

Proof. By computation. In the typed cases using Lemma 2.23 to guarantee derivability and Lemmas 3.17, 3.19 and 3.21 to show that there is no 'bad' occurrences of variables. $\diamond$ $\square$

Let us consider now N and N' of this lemma. Assume that they are mutually inverse isomorphisms, i.e., $N' \circ N$ reduces to identity $[z : K]z$ (and similar thing for $N \circ N'$).

Via two standard β-reductions, we obtain

$$(*)\ \ N' \circ N = [z : K][x_{1\div n} : K_{1\div n}]([x'_{1\div n} : K'_{1\div n}](z M_{1\div n}) M'_{1\div n}).$$

Before we continue with reductions, let us see what can be established about contexts and *kinding* of $M_{1\div n}$ and $M'_{1\div n}$.

To make the notation more compact, let us introduce some abbreviations. Let $1 \leqslant j \leqslant n$. Let μ_1, μ'_1 denote the empty substitution and μ_j, μ'_j $(j > 1)$ denote the substitution $[M_{j-1\div 1}/x_{j-1\div 1}]$, $[M'_{j-1\div 1}/x'_{j-1\div 1}]$, respectively. (Recall that the head variable of M_j is x'_j, and the head variable of M'_j is x_j so the substitutions in the lists do not interact.) Let Ξ_1, respectively, Ξ'_1, denote the empty list, and for $j > 1$ denote the list $x_{1\div j-1} : K_{1\div j-1}$ (respectively, $x'_{1\div j-1} : K'_{1\div j-1}$).

Consider now the kinds of $M_1, \ldots, M_n$ and $M'_1, \ldots, M'_n$. A straightforward use of properties of LF-derivations gives us $(1 \leqslant j \leqslant n)$

$$(1)\ \ \Gamma, z : K, \Xi'_j, x'_j : K'_j \vdash M_j : \mu_j K_j) \textit{ and } (1')\ \ \Gamma, z' : K', \Xi_j, x_j : K_j \vdash M'_j : \mu'_j K'_j,$$

respectively.

Using Corollaries 3.20, 3.21 and then Lemma 2.22 (strengthening), we can make the contexts considerably smaller:

$$(2)\ \ \Gamma, \Xi'_j, x'_j : K'_j \vdash M_j : \mu_j K_j,\ \ (2')\ \ \Gamma, \Xi_j, x_j : K_j \vdash M'_j : \mu'_j K'_j.$$

Of course, from these we can immediately pass to

$$(3)\ \ \Gamma, \Xi'_j \vdash [x'_j : K'_j] M_j : K'_j \to \mu_j K_j\ \ (3')\ \ \Gamma, \Xi_j \vdash [x_j : K_j] M'_j : K_j \to \mu'_j K'_j.$$

As we know, x_j does not occur into K'_j and M'_j, and x'_j does not occur into K_j and M_j, so we have indeed the arrows and not the dependent products at the right.

Of course, if the terms M and M' are mutually inverse, then the terms of each pair must somewhat cancel each other. On the first glance, it seems to pose a problem since only the terms $[x'_1 : K'_1]M_1$ and $[x_1 : K_1]M'_1$ (case of $j = 1$) can be immediately composed, and all other pairs have different contexts. However, if we compute accurately the composition of M and M', we see that the problem dissolves because the mutual invertibility of terms for lesser values of j will make possible other compositions.

Let us consider the following series of typing judgements $(1 \leqslant j \leqslant n)$:

$$(4)\, \Gamma, \Xi_j \vdash [x'_j : \mu'_j K'_j](\mu'_j M_j) : \mu'_j K'_j \to \mu'_j(\mu_j K_j), (4')\, \Gamma, \Xi_j \vdash [x_j : K_j] M'_j : K_j \to \mu'_j K'_j;$$
$$(5.\, \Gamma, \Xi'_j \vdash [x_j : \mu_j K_j](\mu_j M'_j) : \mu_j K_j \to \mu_j(\mu'_j K'_j), 5'.\, \Gamma, \Xi'_j \vdash [x'_j : K'_j] M_j : K'_j \to \mu_j K_j.$$

Notice that in each series the judgements have the same context.

Lemma 3.24. If $\Gamma \vdash M : K \to K', \Gamma \vdash M' : K' \to K$ are mutually inverse isomorphisms in LF, then all the judgements of the series (4), (4′) and (5), (5′) are derivable. Moreover, the jth judgements in (4), (4′) and (5), (5′) represent mutually inverse isomorphisms.

Proof. The judgements in (4′) and (5′) already belong to (3) and (3′) and so are derivable. For the rest, we use a 'truncated' induction on j (up to $j = n$). In case $j = 1$, the substitutions μ and μ' are empty and the derivability is already known. The mutual invertibility follows from the mutual invertibility of M and M'.

To show derivability of (4) and (5) for greater values of j, we take the judgements from (3) and (2′), and proceed using *wkn* and substitution rules of LF (see Remark 2.19). That is, $j-1$ applications of substitution and *wkn* are needed. They are performed one after another using the following schema (we display only the first step).

$$\frac{\dfrac{\Gamma, \Xi'_j \vdash [x'_j : K'_j]M_j : K'_j \to \mu_j K_j}{\Gamma, x_1 : K_1, \Xi'_j \vdash [x'_j : K'_j]M_j : K'_j \to \mu_j K_j} \quad \dfrac{}{\Gamma, x_1 : K_1 \vdash M'_1 : K'_1}}{\Gamma, x_1 : K_1, x'_{2\div j-1} : [M'_1/x'_1]K'_{2\div j-1} \vdash [M'_1/x'_1]([x'_j : K'_j]M_j : K'_j \to \mu_j K_j)}.$$

After that, we proceed up to x_{j-1}. That is, we add $x_2 : K_2, \ldots, x_{j-1} : K_{j-1}$ by *wkn* and substitute $M'_2, \ldots, M'_{j-1}$ for $x'_2, \ldots, x'_{j-1}$ (in this order). Each term is substituted to all kinds and terms to the right, so for each substitution the kinds are adequate. The lemmas about the restrictions of variable occurrences cited above guarantee that there is no substitutions in the 'wrong' places.

As to the mutual invertibility, the induction on j is used to show that $\mu'_j(\mu_j K_j) = K_j$ and $\mu_j(\mu'_j K'_j) = K'_j$ because by induction all the previous terms are mutually inverse isomorphisms (in the corresponding contexts). The mutual invertibility of $[x'_j : \mu'_j K'_j](\mu'_j M_j)$ and $[x_j : K_j]M'_j$, and $[x_j : \mu_j K_j](\mu_j M'_j)$ and $[x'_j : K'_j]M_j$ follow then from the mutual invertibility of M and M'. $\diamond$ □

Theorem 3.25. Let the judgements $\Gamma \vdash M : K \to K'$, and $\Gamma \vdash M' : K' \to K$ be derivable in LF. If the erasure $e(M)$ is a f.h.p., then M is an isomorphism. Moreover, M can be reconstructed up to LF-equality from the kind $K \to K'$ and the term $e(M)$. If the erasures $e(M)$ and $e(M')$ are mutually inverse f.h.p.s, then M and M' are mutually inverse isomorphisms in LF.

Proof. (An outline.) We assume that

$$K \equiv (x_{1\div n} : K_{1\div n})K_0,\ K' \equiv (x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)})K'_0$$

where K_0, K'_0 are either of the form $\mathrm{El}(A)$ or the constant Type. From the structure of the f.h.p., we derive that M has to be of the form

$$M \equiv [z : (x_{1\div n} : K_{1\div n})K_0][x'_{\sigma(1)\div\sigma(n)} : K'_{\sigma(1)\div\sigma(n)}](z M_{1\div n}).$$

Moreover, since σ can be found from $e(M)$ we may use Lemma 3.23 and consider without loss of generality only the case when σ is trivial:

$$M \equiv [z : (x_{1\div n} : K_{1\div n})K_0][x'_{1\div n} : K'_{1\div n}](z M_{1\div n}).$$

The proof of the first implication may proceed by double induction, on the depth of the Bohm tree of the f.h.p., and on j. So we show first that $[x_1 : K'_1]M_1$ is an isomorphism,

and reconstruct it from $\lambda x_1.e(M_1)$. This permits us to find the kind of $[x_2 : K_2']M_2$ as in Lemma 3.24. We proceed with the induction on the second parameter up to n. The depth of the trees of erasures $e([x_i : K_i']M_i) \equiv \lambda x_i.e(M_i)$ is bound by the depth of the tree of $e(M)$.

The second implication is as easy as in Theorems 3.13 and 3.15.$\diamond$ □

3.4. *Examples*

Example 3.26. In the calculus $\lambda^1\beta\eta$, an isomorphism between $A \to B \to C$ and $B \to A \to C$ is given by the term $\lambda z : A \to B \to C.\lambda y : B.\lambda x : A.zxy$. If $B \equiv A$, it becomes an automorphism (different from identity).

Example 3.27. Let $A \equiv \forall X_{1\div 3}.A_4 \to A_5 \to X_2$, where $X_1, X_2, X_3 \notin \mathrm{FV}(A_4) \cup \mathrm{FV}(A_5)$. Let $A' \equiv \forall X_3.(A_5' \to \forall X_1.(A_4' \to \forall X_2.X_2))$. In the notation of Lemma 3.14, $i_1 = 4, i_2 = 5$, $\sigma(1) = 3, \sigma(2) = 5, \sigma(3) = 1, \sigma(4) = 4, \sigma(5) = 2$, $j_1 = 2, j_2 = 4$. If $A \sim A'$, then an isomorphism $M : A \to A'$ may be written as

$$\lambda z : A.\lambda X_3.\lambda x_5 : A_5'.\lambda X_1.\lambda x_4 : A_4'.\lambda X_2.zX_{1\div 3}M_4M_5$$

where $\lambda x_4 : A_4'.M_4 : A_4' \to A_4$ and $\lambda x_5 : A_5'.M_5 : A_5' \to A_5$ are some isomorphisms.

Example 3.28. Consider the following diagram (it is not supposed to be commutative):

$$\begin{array}{ccc}
\forall X_1.\forall X_2.(X_1 \to X_2 \to \forall X.X) & \xrightarrow[M_1]{id} & \forall X_2.\forall X_1.(X_2 \to X_1 \to \forall X.X) \\
{\scriptstyle M_2^{-1}}\uparrow\downarrow{\scriptstyle M_2} & & {\scriptstyle M_2'^{-1}}\uparrow\downarrow{\scriptstyle M_2'} \\
\forall X_2.\forall X_1.(X_1 \to X_2 \to \forall X.X) & \xrightarrow[M_1']{id} & \forall X_1.\forall X_2.(X_2 \to X_1 \to \forall X.X)
\end{array}$$

Notice that the types at the both sides of each horizontal arrow are α-equal, this permits to write *id* there. There exist, though, the isomorphisms different from *id*:

— $M_1 = \lambda z : \forall X_1.\forall X_2.(X_1 \to X_2 \to \forall X.X).\lambda X_2.\lambda X_1.\lambda x_2 : X_2.\lambda x_1 : X_1.zX_1X_2x_1x_2$.
— $M_1' = \lambda z : \forall X_2.\forall X_1.(X_1 \to X_2 \to \forall X.X).\lambda X_1.\lambda X_2.\lambda x_2 : X_2.\lambda x_1 : X_1.zX_2X_1x_1x_2$.
— $M_2 = \lambda z : \forall X_1.\forall X_2.(X_1 \to X_2 \to \forall X.X).\lambda X_2.\lambda X_1.zX_1X_2$.
— $M_2^{-1} = \lambda z : \forall X_2.\forall X_1.(X_1 \to X_2 \to \forall X.X).\lambda X_1.\lambda X_2.zX_2X_1$.
— $M_2' = \lambda z : \forall X_2.\forall X_1.(X_2 \to X_1 \to \forall X.X).\lambda X_1.\lambda X_2.zX_2X_1$.
— $M_2^{-1} = \lambda z : \forall X_1.\forall X_2.(X_2 \to X_1 \to \forall X.X).\lambda X_2.\lambda X_1.zX_1X_2$.

All these terms are normal and different from identity, M_1 and M_1' are automorphisms because the corresponding types are α-equal. Each is inverse to itself. $M_2 =_\alpha M_2'$, $M_2^{-1} =_\alpha M_2'^{-1}$, M_2^{-1} is the inverse of M_2. One may check also that in this example there are no isomorphisms except those listed above and their compositions. This can be verified even without Theorem 3.9, because one can analyse exhaustively the possible normal forms of the λ-terms of the appropriate types.

Example 3.29. The kind $(X_1 : \mathrm{Type})(X_2 : \mathrm{Type})(\mathrm{El}(X_1) \to \mathrm{El}(X_2) \to \mathrm{Type})$ and the kind $(X_2 : \mathrm{Type})(X_1 : \mathrm{Type})(\mathrm{El}(X_2) \to \mathrm{El}(X_1) \to \mathrm{Type})$ are isomorphic. The analysis of possible isomorphisms gives the result similar to example 3.28.

With less trivial dependencies, the structure of isomorphisms becomes more complex.

Example 3.30. Let $\Gamma \vdash (x : K_1)K$ **kind** be derivable in LF. Let $K_1 \sim K_2$ in Γ, and $\Gamma \vdash M : K_1 \to K_2$, $\Gamma \vdash M' : K_2 \to K_1$ be mutually inverse isomorphisms. Then $(x : K_1)K \sim (x' : K_2)[(M'x')/x]K$. The isomorphism from the first to the second kind is given by

$$\Gamma \vdash [z : (x : K_1)K][x' : K_2](z(M'x')) : (x : K_1)K \to (x' : K_2)[(M'x')/x]K,$$

and its inverse by

$$\Gamma \vdash [z : (x' : K_2)[(M'x')/x]K][x : K_1](z(Mx)) : (x' : K_2)[(M'x')/x]K \to (x : K_1)K.$$

If we compose these isomorphisms, M and M' will 'cancel' each other everywhere (inside K as well). Notice that we cannot merely replace K_1 by K_2 (without substitution of M' into K) because the kinds inside K will not match any more. Thus, the 'local' rewrite techniques cannot be used directly to verify whether $K \sim K'$ in difference from the type theories considered in Di Cosmo (1995).

3.5. *The groupoid structure*

Sometimes *groupoids* are defined merely as categories where each arrow is invertible, sometimes the 'smallness' condition is added (Brown et al. 2011). We use this notion only when groupoids are small (even finite), except a very abstract introduction below.

Let K be a category. For $A \in \mathrm{Ob}(K)$ let $K_{\mathrm{iso}}(A)$ denotes the subcategory of K that contains all $A' \in \mathrm{Ob}(K)$ such that $A \sim A'$. Its morphisms are the isomorphisms $f : A' \to A''$, where $A' \sim A \sim A''$. The category $K_{\mathrm{iso}}(A)$ obviously is a groupoid. The graph of $K_{\mathrm{iso}}(A)$ is a connected component of the graph of K.

Let $\mathrm{Aut}_K(A)$ denote the group of automorphisms of A, i.e., isomorphisms $A \to A$. We usually will consider left composition as group multiplication, $fg = f \circ g$. The group $\mathrm{Aut}_K(A)$ may be seen as a category with one object. It is a full subcategory of $K_{\mathrm{iso}}(A)$.

Sometimes we will need the right composition as well. Formally, we may take the opposite category K^{op} and $K^{\mathrm{op}}_{\mathrm{iso}}(A)$, $\mathrm{Aut}_{K^{\mathrm{op}}}(A)$. On the notational side, we shall use; for the right composition, i.e., $f;g = g \circ f$, where $\circ$ is the composition in K. When there is no ambiguity, we shall omit the index K and write $\mathrm{Aut}(A)$ and $\mathrm{Aut}^{\mathrm{op}}(A)$.

Lemma 3.31. Let $A, A' \in \mathrm{Ob}(K)$ be isomorphic. Then (a) $K_{\mathrm{iso}}(A) = K_{\mathrm{iso}}(A')$ and (b) the groups $\mathrm{Aut}_K(A)$ and $\mathrm{Aut}_K(A')$, respectively, $\mathrm{Aut}_{K^{\mathrm{op}}}(A), \mathrm{Aut}_{K^{\mathrm{op}}}(A')$, are isomorphic as groups.

Proof. (a) Obvious. (b) If $f : A \to A'$ and $f^{-1} : A' \to A$ are any mutually inverse isomorphisms, then the isomorphism of groups of automorphisms is defined by the conjugacy with f and f^{-1}. $\diamond$ □

Lemma 3.32. Let $f : A \to A'$ be an isomorphism in K, any other isomorphism $g : A \to A'$ may be uniquely represented as $g = f \circ h = h;f$ and $g = h' \circ f = f;h'$, where $h : A \to A, h' : A' \to A'$.

Proof. We may take $h = f^{-1} \circ g = g; f^{-1}$ and $h' = g \circ f^{-1} = f^{-1}; g$.$\diamond$ □

The following theorem is formulated in full generality, but it will be used only in the situations when no foundational questions (like the use of the axiom of choice) will arise.

Theorem 3.33. For each pair $B, C \in \mathrm{Ob}(K_{\mathrm{iso}}(A))$, a distinguished isomorphism f_{BC} may be selected in such a way that every diagram of distinguished isomorphisms commutes. The result obviously holds also for the isomorphisms in $K^{\mathrm{op}}_{\mathrm{iso}}(A)$.

Proof. Let us fix an isomorphism $f_{AD} : A \to D$ for each $D \in \mathrm{Ob}(K_{\mathrm{iso}}(A))$, define $f_{DA} = f^{-1}_{AD}$ and $f_{BC} = f_{AC} \circ f^{-1}_{AB}$ when B, C are different from A. The proof for $K^{\mathrm{op}}_{\mathrm{iso}}(A)$ is similar. $\diamond$ □

Remark 3.34. In terms of Luo et al. (2013), it means that these distinguished isomorphisms form a coherent system of basic coercions and may be used to define coercive subtyping extensions of the corresponding type theories.

4. Automorphisms of types in $\lambda^1\beta\eta\pi*$

In this section, we follow very closely the book (Di Cosmo 1995). The main difference is that the results of Di Cosmo (1995) establish only the link between the *existence* of an isomorphism and the *existence* of a derivation in a certain formal theory.

For example, the theorems in Di Cosmo (1995) about soundness and completeness of the theory of isomorphism of types state that two types A, B are isomorphic according to the definition based on *existence* of mutually inverse terms $\vdash M : A \to B$ and $\vdash M^{-1} : B \to A$ if and only if they are equal in a certain theory based on the rewrite rules, such as *swap*:

$$A \to B \to C = B \to A \to C.$$

In the difference from Di Cosmo, we are interested in the algebraic structure of the set of isomorphisms and its subset that consists of automorphisms, and the *existence* of an isomorphism does not permit to describe this algebraic structure as a whole.

As we already noticed, the calculus $\lambda^1\beta\eta\pi*$ may be seen as a (free) cartesian closed category (cf. Di Cosmo (1995), p.102). Its objects are types, and morphisms $A \to B$ are the equivalence classes of λ-terms of the type $A \to B$ w.r.t. the equivalence based on the $\beta\eta$-reduction described above.

The calculus $\lambda^1\beta\eta$ may be seen as its subcategory.

Lemma 4.1. Let A, B be types of $\lambda^1\beta\eta$ (i.e., they do not contain $\wedge$ and T), and $\vdash M : A \to B$ be term of $\lambda^1\beta\eta\pi*$. Then the normal form of M belongs to $\lambda^1\beta\eta$.

Proof. This lemma is a combination of Lemmas 4.2.2 and 4.2.6 of Di Cosmo (1995).$\diamond$ □

4.1. *Automorphisms in* $\lambda^1\beta\eta$

In this subsection, all types belong to $\lambda^1\beta\eta$ and the groups $\mathrm{Aut}(A), \mathrm{Aut}^{\mathrm{op}}(A)$ are considered w.r.t. the category $K = \lambda^1\beta\eta$. For any group G, G^{op} will denote the same group with

reverse multiplication; of course, $(G^{\mathrm{op}})^{\mathrm{op}} = G$. One may notice also that G and G^{op} are isomorphic (with $g \mapsto g^{-1}$ as an isomorphism).

The necessary notions of the theory of groups, such as wreath product, may be found in Hall (1959), see also Pólya (1937), Harari (1969) and White (1984).

The following theorem is central for this section. Let Σ_m denotes the symmetric group of m elements, and $\Sigma_m | \wr G$ the wreath product of Σ_m and G.

Theorem 4.2. The groups $\mathrm{Aut}(A)$ (and $\mathrm{Aut}^{\mathrm{op}}(A)$) are, up to group isomorphisms, exactly the groups that that belong to the class W of finite groups defined inductively as follows:

a. $\{1\} \in W$.
b. If $G, H \in W$, then $G \times H \in W$.
c. If $G \in W$ and $m \geqslant 2$, then $\Sigma_m | \wr G$.

Proof. Let A be a type of $\lambda^1\beta\eta$. Let us prove by induction on $d(A)$ that the group $\mathrm{Aut}(A)$ is isomorphic to a group $G \in W$. (It follows immediately that $\mathrm{Aut}^{\mathrm{op}}(A)$ is isomorphic to G as well.) □

Base case: $d(A) = 0$, $A \equiv p$. This case is obvious.

Inductive step. Let $d(A) = n + 1$, $A \equiv A_1 \to \ldots A_n \to p$. Let $M : A \to A$ be an automorphism. By Lemma 3.12, we may assume that $M \equiv \lambda z : A.\lambda x_{\sigma(1)\div\sigma(n)} : A_{\sigma(1)\div\sigma(n)}.zM_{1\div n}$. The types should match, thus we may also assume that (i) $A_{\sigma(i)} \equiv A_i$, and (ii) the terms $\lambda x_i : A_i.M_i$ have $A_i \to A_i$ as types and are automorphisms as well.

Since the automorphism groups of isomorphic types are isomorphic, we may assume without loss of generality that the isomorphic types among $A_1, \ldots, A_n$ are in fact identical, and are placed (using *swap*) in such a way that

$$A_1 \equiv \ldots \equiv A_{i_1} \nsim A_{i_1+1} \equiv \ldots \equiv A_{i_1+i_2} \nsim \ldots \nsim A_{i_1+\ldots+i_{k-1}+1} \equiv \ldots \equiv A_{i_1+\ldots+i_k}$$

where $i_1 + \ldots + i_k = n$. The relation $\nsim$ means that the types are *not* isomorphic. Given the isomorphisms for $A_i's$, an isomorphism between an arbitrary type $A_1 \to \ldots A_n \to p$ and a type that satisfies these assumptions can be in fact written explicitly.

Let $G_1 = \mathrm{Aut}(A_1)$, $G_2 = \mathrm{Aut}(A_{i_1+1}), \ldots$, $G_k = \mathrm{Aut}(A_{i_1+\ldots+i_{k-1}+1})$, the $G_1^{\mathrm{op}}, \ldots, G_k^{\mathrm{op}}$ the same groups with reverse multiplication and Σ_m denote the symmetric group acting on m elements. By inductive hypothesis, the groups $G_1, \ldots, G_k$ and $G_1^{\mathrm{op}}, \ldots, G_k^{\mathrm{op}}$ are isomorphic to some groups that belong to the class W.

Remark 4.3. To proceed correctly with induction, we need to consider both the groups with left and right compositions as group operations, because $\to$ (as a functor) is contravariant on the first argument.

The wreath products $H_j = \Sigma_{i_j} | \wr G_j$ are defined in a usual way. That is, their elements are pairs $(\sigma, (R_1, \ldots, R_{i_j}))$, and the product of two elements in H_j

$$(\sigma, (R_1, \ldots, R_{i_j}))(\rho, (S_1, \ldots, S_{i_j})) =_{\mathrm{def}} (\sigma\rho, (R_{\rho(1)} \circ S_1, \ldots, R_{\rho(i_j)} \circ S_{i_j})).$$

We define also $H'_j = \Sigma_{i_j} | \wr G_j^{\mathrm{op}}$. Notice that if $i_j = 1$, then $H_j = G_j$, $H'_j = G_j^{\mathrm{op}}$.

The cartesian product $H = H_1 \times \ldots \times H_k$ (respectively, $H' = H'_1 \times \ldots \times H'_k$) consists of k-tuples of pairs with componentwise multiplication described above. To lighten the

notation, we shall use a different representation for H and H' (strictly speaking, it is connected with the representation that uses k-tuples by an obvious isomorphism). Namely, instead of the k-tuple

$$((\sigma_1,(R_1^1,\ldots,R_{i_1}^1)),\ldots,(\sigma_k,(R_1^k,\ldots,R_{i_k}^k)),$$

we shall take $(\sigma,(R_1,\ldots,R_n))$, where $n=i_1+\ldots+i_k$, $\sigma=\sigma_1+\ldots+\sigma_k$ and $R_1=R_1^1,\ldots,R_{i_1}=R_{i_1}^1,R_{i_1+1}=R_1^2,\ldots,R_{i_1+i_2}=R_{i_2}^2,\ldots,R_{i_1+\ldots+i_{k-1}+1}=R_1^k,\ldots,R_n=R_{i_k}^k$. Since the direct sum of permutations acts independently on each 'cluster,' the multiplication can be defined in the same way as for wreath product above, that is

$$(\sigma,(R_1,\ldots,R_n))(\rho,(S_1,\ldots,S_n))=(\sigma\rho,(R_{\rho(1)}\circ S_1,\ldots,R_{\rho(n)}\circ S_n))$$

(with ; instead of $\circ$ in case of H').

Let M and N be automorphisms of A. As explained in the beginning of the proof, we may assume that $M:A\to A$ and $N:A\to A$ have the form:

$$M\equiv\lambda z:A.\lambda x_{\sigma(1)\div\sigma(n)}:A_{1\div n}.zM_{1\div n},$$
$$N\equiv\lambda z:A.\lambda x_{\rho(1)\div\rho(n)}:A_{1\div n}.zN_{1\div n}.$$

Let us consider the compositions $M;N=N\circ M\equiv\lambda z:A.(N(Mz))$ and $M\circ N\equiv\lambda z:A.(M(Nz))$.

To shorten the notation, we omit the types $A_{1\div n},A$. From $M;N=N\circ M$ by two β-reductions, we obtain

$$\lambda z.\lambda x_{\rho(1)\div\rho(n)}.(\lambda x_{\sigma(1)\div\sigma(n)}.zM_{1\div n})N_{1\div n}.$$

Now by β-reductions corresponding to $\lambda x_{\sigma(1)\div\sigma(n)}$, we get

$$\lambda z.\lambda x_{\rho(1)\div\rho(n)}.z([N_{\sigma^{-1}(1)}/x_1]M_1)\ldots([N_{\sigma^{-1}(n)}/x_n]M_n).$$

The terms $M_i,N_{\sigma^{-1}(i)}$ contain only one free variable. In the term $N_{\sigma^{-1}(i)}$ and, as consequence, in the result of substitution, it is the variable $x_{\sigma^{-1}(i)}$.

If we rename the variables by α-conversion: $x_i\mapsto x_{\sigma(i)}$, the result will be

$$\lambda z.\lambda x_{\sigma(\rho(1))\div\sigma(\rho(n))}.z([[x_1/x_{\sigma^{-1}(1)}]N_{\sigma^{-1}(1)}/x_1]M_1)...([[x_n/x_{\sigma^{-1}(n)}]N_{\sigma^{-1}(n)}/x_n]M_n).$$

We notice also that the term $\lambda x_{\sigma^{-1}(i)}.[N_{\sigma^{-1}(i)}/x_i]M_i$ is β-equal (by β-reduction from right to left) to

$$\lambda x_{\sigma^{-1}(i)}.(\lambda x_i.M_i((\lambda x_{\sigma^{-1}(i)}.N_{\sigma^{-1}(i)})x_{\sigma^{-1}(i)})$$

and this by definition is the composition

$$(\lambda x_{\sigma^{-1}(i)}.N_{\sigma^{-1}(i)});(\lambda x_i.M_i)=(\lambda x_i.M_i)\circ(\lambda x_{\sigma^{-1}(i)}.N_{\sigma^{-1}(i)}).$$

In both cases, $\lambda x_{\sigma^{-1}(i)}.N_{\sigma^{-1}(i)}=_\alpha\lambda x_i.[x_i/x_{\sigma^{-1}(i)}]N_{\sigma^{-1}(i)}$.

Remark 4.4. According to this analysis, the inverse

$$M^{-1}=\lambda z.\lambda x_{\sigma^{-1}(1)\div\sigma^{-1}(n)}.zM'_{1\div n},$$

where $\lambda x_i.M_i'=(\lambda x_{\sigma(i)}.M_{\sigma(i)})^{-1}\ \ (1\leqslant i\leqslant n)$.

Remark 4.5. For both automorphisms M and N, the permutations σ and ρ act independently on the indexes of each 'cluster' of identical premises, and so may be uniquely represented as $\sigma_1 + ... + \sigma_k$ and $\rho_1 + \ldots + \rho_k$ with $\sigma_1, \rho_1 \in \Sigma_{i_1}, \ldots, \sigma_k, \rho_k \in \Sigma_{i_k}$.

Let us consider ; as a group multiplication on $A \to A$. We define the isomorphism $\theta : \mathrm{Aut}(A) \to H$ as follows. For

$$M \equiv \lambda z : A.\lambda x_{\sigma(1)\div\sigma(n)} : A_{1\div n}.zM_{1\div n},$$

we define

$$\theta(M) = (\sigma, (R_1, \ldots, R_n)),$$

where $R_i = \lambda x_{\sigma(i)}.M_{\sigma(i)}$ $(1 \leqslant i \leqslant n)$.

It is obvious that θ preserves the group unit. It is a bijection since the groups G_j are the corresponding automorphism groups.

The inverse $(\sigma^{-1}, (R^{-1}_{\sigma^{-1}(1)}, \ldots, R^{-1}_{\sigma^{-1}(n)}))$ coincides with $\theta(M^{-1})$ (see Remark 4.4).

Let us verify that it respects multiplication. Let

$$\theta(M) = ((\sigma, (R^1_1, \ldots, R_n)) \text{ and } \theta(N) = ((\rho, (S_1, \ldots, S_n)).$$

The permutation that corresponds to $M;N$ is $\sigma\rho$, and under $\sigma\rho$ the 'clusters' remain invariant. Here, R_i are as above, and $S_i = \lambda x_{\rho(i)}.N_{\rho(i)}$ $(1 \leqslant i \leqslant n)$. In H

$$\begin{aligned}
&\theta(M)\theta(N) = ((\sigma, (R_1, \ldots, R_n))((\rho, (S_1, \ldots, S_n)) = \\
&(\sigma\rho, (R_{\rho(1)} \circ S_1, \ldots, R_{\rho_1(n)} \circ S_n)) = \\
&(\sigma\rho, ((\lambda x_{\sigma(\rho(1))}.M_{\sigma(\rho(1))}) \circ (\lambda x_{\rho(1)}.N_{\rho(1)}), \ldots, (\lambda x_{\sigma(\rho(n))}.M_{\sigma(\rho(n))}) \circ (\lambda x_{\rho(n)}.N_{\rho(n)}))) = \\
&(\sigma\rho, ((\lambda x_{\sigma(\rho(1))}.M_{\sigma(\rho(1))}) \circ (\lambda x_{\sigma^{-1}(\sigma(\rho(1)))}.N_{\sigma^{-1}(\sigma(\rho(1)))}), \ldots, \\
&(\lambda x_{\sigma(\rho(1))}.M_{\sigma(\rho(1))}) \circ (\lambda x_{\sigma^{-1}(\sigma(\rho(n)))}.N_{\sigma^{-1}(\sigma(\rho(n)))}))) = \theta(M;N)
\end{aligned}$$

Consider now $\circ$ as group multiplication in $A \to A$. The isomorphism $\theta' : \mathrm{Aut}^{\mathrm{op}}(A) \to H'$ is defined by $\theta'(M) = (\sigma^{-1}, (\lambda x_1.M_1, \ldots, \lambda x_n.M_n))$ for $M : A \to A$ as above.

It is easily seen that θ' preserves unit and inverse, and is bijective. As to the multiplication, we have

$$\begin{aligned}
&(\theta'(N))(\theta'(M)) = (\rho^{-1}, (\lambda x_1.N_1, \ldots, \lambda x_n.N_n))(\sigma^{-1}, (\lambda x_1.M_1, \ldots, \lambda x_n.M_n)) = \\
&(\rho^{-1}\sigma^{-1}, ((\lambda x_{\sigma^{-1}(1)}.N_{\sigma^{-1}(1)}; \lambda x_1.M_1), \ldots, (\lambda x_{\sigma^{-1}(n)}.N_{\sigma^{-1}(n)}; \lambda x_n.M_n))) = \\
&((\sigma\rho)^{-1}, (\lambda x_{\sigma^{-1}(1)}.[N_{\sigma^{-1}(1)}/x_1]M_1, \ldots, \lambda x_{\sigma^{-1}(n)}.[N_{\sigma^{-1}(n)}/x_n]M_n)) = \theta'(N \circ M).
\end{aligned}$$

Let us consider now some $G \in W$. The type A such that G is isomorphic to $\mathrm{Aut}(A)$ is constructed by induction on the process of construction of G. Let us notice that the group $\mathrm{Aut}(B)$ is isomorphic to $\mathrm{Aut}(A)$ if B is obtained from A by renaming of variables, and the same for $\mathrm{Aut}^{\mathrm{op}}$.

Base case (a) $G = \{1\}$. We may take $A \equiv p$.

Inductive step.

(b) Let $G = G_1 \times G_2$, and G_i be isomorphic to $\mathrm{Aut}(A_i)$ $(i = 1, 2)$. Without loss of generality, we may assume that A_1, A_2 have no common type variables. The group $\mathrm{Aut}(A_1 \to A_2 \to p)$ with p fresh will be isomorphic to $G_1^{\mathrm{op}} \times G_2^{\mathrm{op}}$, and thus to $G_1 \times G_2$. We may take also $\mathrm{Aut}((A_1 \to p_1) \to (A_2 \to p_2) \to p)$ with p_1, p_2, p fresh.

(c) Let $G = \Sigma_m \wr G_0$. By induction, there exists A_0 such that $\mathrm{Aut}(A_0)$ is isomorphic to G_0. If we take $A \equiv A_0 \to \ldots A_0 \to p$ (m identical premises, p fresh), the group $\mathrm{Aut}(A)$ will be isomorphic to $\Sigma_m \wr G_0^{\mathrm{op}}$ and (because G_0 is isomorphic to G_0^{op}) to $\Sigma_m \wr G_0$ as well, we may also take $(A_0 \to p_0) \to \ldots (A_0 \to p_0) \to p$ as A (with p_0 and p fresh). $\diamond$

There is a well-known result that the groups of automorphisms of finite trees are exactly the groups of the class W, see Babai (1995), p. 1457, Proposition 1.15. (In fact, as notices Babai, this result was first obtained by Jordan in XIX century.)

As a consequence, we obtain the next corollary:

Corollary 4.6. The class of groups $\mathrm{Aut}(A)$ (considered up to group isomorphisms) where A are types of $\lambda^1\beta\eta$ coincides with the class of automorphisms of finite trees.

4.2. *Reduction of the $\lambda^1\beta\eta\pi*$-case to the case of $\lambda^1\beta\eta$*

In this subsection, the types belong to $\lambda^1\beta\eta\pi*$ and the groups $\mathrm{Aut}(A), \mathrm{Aut}^{\mathrm{op}}(A)$ are considered w.r.t. the category $K = \lambda^1\beta\eta\pi*$. As we shall see, it will not change the class of groups representable as the automorphism groups of types.

Definition 4.7. (Cf. Di Cosmo (1995), p. 104.) A type B in $\lambda^1\beta\eta\pi^*$ is in type normal form (abbreviated $t-nf$) if B is either T or $B \equiv B_1 \times \ldots \times B_n$ where each B_i does not contain $\times$ or T.

Remark 4.8. Notice that every type in $\lambda^1\beta\eta$ is in $t-nf$.

Di Cosmo considered the type-reduction relation $\mathcal{R}^1$ generated by

$$A \times (B \times C) > (A \times B) \times C; \quad (A \times B) \to C > A \to (B \to C);$$
$$A \to (B \times C) > (A \to B) \times (A \to C); \quad A \times T > A; \quad T \times A > A;$$
$$A \to T > T; \quad T \to A > A.$$

It transforms every type into another isomorphic type.

Proposition 4.9. (Di Cosmo (1995), proposition 4.1.2.) Each type in $\lambda^1\beta\eta\pi^*$ has a unique normal form in $\mathcal{R}^1$.

(This normal form may be called the $t-nf$ of A.)

Lemma 4.10. Let B be the $t-nf$ of $A \to p$. Then B does not contain $\times$ and T, and thus belongs to $\lambda^1\beta\eta$.

Proof. By induction on the length of a reduction sequence. $\diamond$ □

Lemma 4.11. Let A be any type in $\lambda^1\beta\eta\pi*$. The group $\mathrm{Aut}(A)$ is isomorphic as group to $\mathrm{Aut}^{\mathrm{op}}(A \to p)$ where p is a type variable that does not occur in A (respectively, $\mathrm{Aut}^{\mathrm{op}}(A)$ is isomorphic to $\mathrm{Aut}(A \to p)$).

Proof. Let $M : A \to A$ be a closed term representing an automorphism of A in $\lambda^1\beta\eta\pi*$. Let us define $\phi(M) \equiv \lambda y : A \to p.\lambda x : A.y(Mx) \equiv [M/z](\lambda y : A \to p.\lambda x : A.y(zx))$. This

transformation preserves identity and reverses the order of composition. For example,

$$\lambda z : A \to p.(\lambda y : A \to p.\lambda x : A.y(Mx))(\lambda y : A \to p.\lambda x : A.y(M^{-1}x))z) =_1,$$
$$\lambda z : A \to p.(\lambda y : A \to p.\lambda x : A.y(Mx))(\lambda x : A.(z(M^{-1}x))) =_1,$$
$$\lambda z : A \to p.(\lambda x : A.(\lambda x : A.z(M^{-1}x))(Mx)) =_1,$$
$$\lambda z : A \to p.(\lambda x : A.z(M^{-1}(Mx))) =_1 \lambda z : A \to p.(\lambda x : A.zx) =_1 id_{A\to p}.$$

Let us show that ϕ is bijective.

Let $M \neq_1 M'$. It implies $Mx \neq_1 M'x$. Notice that at least one is not equal to id_A. If both are different from id_A, then the (unique) normal forms $nf(Mx)$ and $nf(M'x)$ are different from x, and the (unique) $nf(\lambda y : A \to p.\lambda x : A.y(Mx)) \equiv \lambda y : A \to p.\lambda x : A.y(nf(Mx))$ is different from $nf(\lambda y : A \to p.\lambda x : A.y(M'x)) \equiv \lambda y : A \to p.\lambda x : A.y(nf(M'x))$.

If one is equal to id_A, say $M' =_1 \lambda x : A.x$, then $nf(\lambda y : A \to p.\lambda x : A.y(Mx)) \equiv \lambda y : A \to p.\lambda x : A.y(nf(Mx))$ is different from $nf(\lambda y : A \to p.\lambda x : A.y(M'x)) =_1 \lambda y : A \to p.\lambda x : A.yx =_1 \lambda y : A \to p.y$.

Thus, ϕ is injective. Let us show that it is bijective.

Consider any automorphism $M : (A \to p) \to A \to p$. The results that concern erasure and f.h.p.s are not directly applicable to the calculus with pairing, such as $\lambda^1\beta\eta\pi*$, but the proof below is based on the same kind of ideas.

Without loss of generality, we may assume that the closed term M is normal. By normality, it has to be of the form $\lambda y_{1\div n} : A_{1\div n}.yM_{1\div k}$. The inverse automorphism should have a similar form.

Since the type is $(A \to p) \to A \to p$, the λ-prefix is either $\lambda y_1 : A \to p$ or $\lambda y_1 : A \to p.\lambda y_2 : A$. The structure of these types and the absence of free variables imply that in the first case y must coincide with y_1 and the term

$$\lambda y_1 : A \to p.y_1 =_1 \lambda y_1 : A \to p.\lambda y_2 : A.y_1y_2 =_1,$$
$$\lambda y_1 : A \to p.\lambda y_2 : A.y_1((\lambda x : A.x)y_2) \equiv \phi(id_A).$$

By the same reasons, in the second case, y still has to coincide with y_1, and y_2 be of the type A; the term (normal) now must have the structure $\lambda y_1 : A \to p.\lambda y_2 : A.y_1M_1$ with M_1 normal and different from y_2. The type of M_1 has to be A.

The analysis of the structure of M^{-1} shows that it has to have similar structure. In the second case, it has to be $\lambda y'_1 : A \to p.\lambda y'_2 : A.y'_1M'_1$. Since the composition of M and M^{-1} must give $id_{A\to p}$, we easily derive that $\lambda y_1 : A.M_1$ and $\lambda y'_1 : A.M'_1$ are mutually inverse automorphisms, and $\phi(M_1) = M$, $\phi(M'_1) = M^{-1}$. $\diamond$ □

As a consequence, we have

Theorem 4.12. The groups $\mathrm{Aut}(A)$ (and $\mathrm{Aut}^{\mathrm{op}}(A)$) where A are types of $\lambda^1\beta\eta\pi*$ are, up to group isomorphisms, exactly the groups that may be obtained from symmetric groups by the operations of cartesian product and wreath product. This class coincides with the class of automorphism groups of finite trees.

Remark 4.13. One may notice that the calculus $\lambda^1\beta\eta$ (the term system as well) may be seen also as the $\multimap$-fragment of the Intuitionistic Multiplicative Linear Logic

(Soloviev 1993), and our Theorem 4.2 and Corollary 4.6 will hold there. Theorem 4.12 will probably hold for full IMLL instead of $\lambda^1\beta\eta\pi*$ but reduction techniques need to be modified because of the absence of the isomorphism $A \to (B \otimes C) > (A \to B) \otimes (A \to C)$ and more complex relationship between term systems.

5. Automorphisms of higher order types

In this section, we will show that any finite group may be represented by the group of automorphisms of (a) second-order types and (b) dependent products.

The role of a 'bridge' between algebra and type theory will be played by the groups of automorphisms of graphs. It is well known since the works of Frucht (1938, 1949) that finite groups can be represented by automorphism groups of finite graphs (not coloured). Below, an older but more straightforward approach is taken: Our construction is based on the construction of the Cayley-coloured digraph. We will

— construct a type for each Cayley-coloured digraph,
— for each automorphism of this digraph construct a λ-term that represents an automorphism of that type,
— and show that the correspondence is a group isomorphism.

If G is a group with elements $g_1, \ldots, g_n$ (we shall assume that g_1 is the group unit) and $S = \{s_1, \ldots, s_m\} \subset \{g_1, \ldots, g_n\}$ some set of its (distinct, non-trivial) generators, then the Cayley graph G for S is a directed graph $C_S(G)$ where the directed edges are assigned colours. The nodes of $C_S(G)$ are the elements of $\{1, \ldots, n\}$. To the generators are associated different colours $c_1, \ldots, c_m$. There is an edge from i to j coloured c_k iff $g_i \cdot s_k = g_j$.

If H is a digraph on $\{1, \ldots, n\}$, digraph automorphisms of H are defined by permutations θ of $1, \ldots, n$ that respect directed adjacences, i.e., (i, j) is a directed edge in H iff $(\theta(i), \theta(j))$ is also a directed edge. The group under composition of digraph automorphisms of H is denoted $\mathrm{Aut}(H)$.

If H is edge-coloured, then $\theta \in \mathrm{Aut}(H)$ is colour-preserving if the colour of $(\theta(i), \theta(j))$ is the same as the colour of (i, j) for all edges (i, j) of H.

The automorphisms that preserve colour form a subgroup of $\mathrm{Aut}(H)$.

Theorem 5.1. (See, for example, White (1984), Theorems 4–8.) The subgroup of colour-preserving automorphisms of $\mathrm{Aut}(C_S(G))$ is isomorphic to G. The isomorphism of G and this subgroup may be defined by $g \mapsto \theta_g$ where the permutation θ_g of the set $\{1, \ldots, n\}$ is defined by left multiplication, $\theta_g(i) = j \iff gg_i = g_j$.

Remark 5.2. Obviously, $\theta_{g_1} = id$, $\theta_{g^{-1}} = \theta_g^{-1}$ and $\theta_{g'g} = \theta_{g'}\theta_g$.

Below, we shall denote by σ_k the permutation of the set $\{1, \ldots, n\}$ defined by the *right* multiplication of $\{g_1, \ldots, g_n\}$ by the generator s_k, i.e., $\sigma_k(i) = j$ iff $g_i s_k = g_j$.

We shall built types such that their groups of type automorphisms are isomorphic to groups of automorphisms of Cayley-coloured graphs.

5.1. *Second-order λ-calculus*

Let some finite group G with elements $|G| = \{g_1, \ldots, g_n\}$ and generators $S = \{s_1, \ldots, s_m\}$ be given. We shall construct the closed type $T_S(G)$ in $\lambda^2\beta\eta$ such that the group $\mathrm{Aut}(T_S(G))$ is isomorphic (as a group) to G.

Remark 5.3. Since Di Cosmo (1995) established that every isomorphism $M : A \to A'$ in $\lambda^2\beta\eta\pi*$, where A, A' belong to $\lambda^2\beta\eta$ is equal to an isomorphism in $\lambda^2\beta\eta$, our construction automatically gives a similar result for $\lambda^2\beta\eta\pi*$.

Let us consider the types $C_1^- \equiv Y \to Y$, $C_2^- \equiv C_1^- \to Y, \ldots, C_m^- \equiv C_{m-1}^- \to Y$ and $C_1 \equiv \forall Y.C_1^-, \ldots, C_m \equiv \forall Y.C_m^-$ $(d(C_i) = i)$.

Lemma 5.4. Let $X_1, \ldots, X_n$ be distinct type variables. The types $(X_i \to X_j) \to C_k$ and $(X_{i'} \to X_{j'}) \to C_{k'}$ are isomorphic iff $i = i', j = j', k = k'$, and in this case the only automorphism is identity.

Proof. We use Lemma 3.14 for the structure of isomorphism terms.

First, we show by induction on $\min(k, k')$ that if $k \neq k'$, then $C_k^- \nsim C_{k'}^-$ and $C_k \nsim C_{k'}$, and the only automorphism of C_k^- or C_k is identity. As a trivial exercise, we show also that $(X_i \to X_j) \sim (X_{i'} \to X_{j'})$ iff $i = i'$ and $k = k'$ and in this case the only automorphism is identity as well.

If $i = i', j = j'$ and $k = k'$, we notice that by the same lemma any automorphism of $(X_i \to X_j) \to C_k$ will use an automorphism of $(X_i \to X_j)$ and of C_k^-, and conclude that the only automorphism is identity. $\diamond$ □

The type $T_S(G)$ is defined as follows. We use n type variables $X_1, \ldots, X_n$ to represent $g_1, \ldots, g_n$, and the types $C_1, \ldots, C_m$ to represent 'colours.'

For each directed edge (i, j) of the Cayley-coloured graph $C_S(G)$ where $g_i s_k = g_j$, that is $j = \sigma_k(i)$, we take the type $T_{ik} = (X_i \to X_j) \to C_k$.

We define

$$T_S(G) \equiv \forall X_{1 \div n}.T_{11} \to \ldots T_{1m} \to T_{21} \to \ldots T_{2m} \to \ldots T_{n1} \to \ldots T_{nm} \to \forall Y.Y.$$

Theorem 5.5. (Main.) The group of automorphisms of the type $T_S(G)$ is isomorphic to the group of automorphisms of $C_S(G)$ and thus isomorphic to G.

Proof. Let us consider an automorphism θ_g of $C_S(G)$ that preserves colour. To define the term $M_{\theta_g} : T_S(G) \to T_S(G)$, we proceed as follows. (Below, we do *not* always rename variables related to different binders because it makes easier to follow the proof 'globally.')

— First, we notice that the type

$$T'_S(G) \equiv \forall X_{\theta_g(1) \div \theta_g(n)}.T_{\theta_g(1)1} \to \ldots T_{\theta_g(1)m} \to T_{\theta_g(2)1} \to \ldots T_{\theta_g(2)m} \to \ldots$$
$$T_{\theta_g(n)1} \to \ldots T_{\theta_g(n)m} \to \forall Y.Y.$$

is α-equal to

$$T_S(G) \equiv \forall X_{1 \div n}.T_{11} \to \ldots T_{1m} \to T_{21} \to \ldots T_{2m} \to \ldots T_{n1} \to \ldots T_{nm} \to \forall Y.Y.$$

Indeed, we may apply the α-conversion where the renaming is defined by the inverse permutation θ_g^{-1}:

$$X_{\theta_g(1)} \mapsto X_1, \ldots, X_{\theta_g(n)} \mapsto X_n.$$

— We consider the following term of type $T_S(G) \to T'_S(G)$:

$$\begin{aligned}
&\lambda z : T_S(G).\lambda X_{\theta_g(1)\div\theta_g(n)}.\\
&\lambda x_{\theta_g(1)1\div\theta_g(1)m} : T_{\theta_g(1)1\div\theta_g(1)m} \ldots \lambda x_{\theta_g(n)1\div\theta_g(n)m} : T_{\theta_g(n)1\div\theta_g(n)m}.\\
&zX_{1\div n}x_{11\div 1m}\ldots x_{n1\div nm}.
\end{aligned}$$

— It is an isomorphism, with the inverse given explicitly by

$$\begin{aligned}
&\lambda z : T'_S(G).\lambda X_{\theta_g^{-1}(1)\div\theta_g^{-1}(n)}.\\
&\lambda x_{\theta_g^{-1}(1)1\div\theta_g^{-1}(1)m} : T_{\theta_g^{-1}(1)1\div\theta_g^{-1}(1)m} \ldots \lambda x_{\theta_g^{-1}(n)1\div\theta_g^{-1}(n)m} : T_{\theta_g^{-1}(n)1\div\theta_g^{-1}(n)m}.\\
&zX_{1\div n}x_{11\div 1m}\ldots x_{n1\div nm}.
\end{aligned}$$

(it can be verified by direct computation).

— It is α-equivalent to the term

$$\begin{aligned}
&M_{\theta_g} \equiv \lambda z : T_S(G).\lambda X_{1\div n}.\lambda x_{11\div 1m} : T_{11\div 1m} \ldots \lambda x_{n1\div nm} : T_{n1\div nm}.\\
&zX_{\theta^{-1}(1)\div\theta^{-1}(n)}x_{(\theta_g^{-1}(1)1\div(\theta_g^{-1}(1))m}\ldots x_{(\theta_g^{-1}(n))1\div(\theta_g^{-1}(n))m}).
\end{aligned}$$

The fact that $\theta_g \mapsto M_{\theta_g}$ defines a group homomorphism $C_S(G) \to \mathrm{Aut}(T_S(G)$ is verified now by direct computation (using β, η and α).

The map $\theta_g \mapsto M_{\theta_g}$ is injective because all terms $M_{\theta_g}, g \neq g_1$, are normal (since multiplication by g has no fixpoints) and different from each other and not identity. The term $M_{\theta_{g_1}} = id_{T_S(G)}$.

The last point we need to verify is that the type $T_S(G)$ does not have automorphisms that are not of the form M_{θ_g}, hence the map we have defined is surjective.

We do it in several steps.

Let M be some term of type $T_S(G) \to T'$.

i. If M is an automorphism, T' has to be α-equal to $T_S(G)$. What are the immediate consequences?

— First of all, $T' \equiv \forall X_{\theta(1)\div\theta(n)}.(T'_{\rho(11)} \to \ldots \to T'_{\rho(nm)} \to \forall Y.Y)$, where ρ is some permutation of pairs $11,\ldots,nm$. The place and length of the $\forall$-prefix cannot change. Also, for each i,k, it should be possible to obtain the type T_{ik} from $T'_{\rho(ik)}$ by a renaming of variables.

— It implies that ρ must respect 'colours' C_k, because variable renamings cannot change the size of (the number of variable occurrences in) C_k. Thus, $T'_{\rho(ik)} \equiv (X_{i'} \to X_{j'}) \to C_k$, $i', j' \in \{1,\ldots,n\}$.

— The permutation θ in the $\forall$-prefix determines the unique renaming that may transform T' into $T_S(G)$: It has to be θ^{-1}, $X_{\theta(i)} \mapsto X_i$ because the prefix has to become $\forall X_{1\div n}$.

— Moreover, it implies that $T'_{\rho(ik)} \equiv (X_{\theta(i)} \to X_{\theta(\sigma_k(i))}) \to C_k$, else it will not become T_{ik} after renaming.

ii. As a next step, we shall take into account that an automorphism is an isomorphism. By Lemma 3.14, we may assume that

$$M \equiv \lambda z : T_S(G).\lambda X_{\theta(1)\div\theta(n)}.\lambda x_{\rho(11)\div\rho(nm)} : T'_{\rho(11)\div\rho(nm)}.zX_{1\div n}M_{11\div nm},$$

where the terms $\lambda x_{ik} : T'_{ik}.M_{ik}$ are isomorphisms.

Let $\rho(i'k') = ik$. By (i), $T'_{\rho(i'k')} \equiv (X_{\theta(i')} \to X_{\theta(\sigma_k(i'))}) \to C_k$. By Lemma 5.4, an isomorphism between $T'_{\rho(i'k')}$ and T_{ik} exists only when $T'_{\rho(i'k')} \equiv T_{ik}$. If it exists, it must be identity, i.e., $M_{ik} \equiv x_{ik}$.

If $X_{\theta(i')} \equiv X_i$ then $i = \theta^{-1}(i')$ and $T'_{ik} \equiv (X_i \to X_{\theta(\sigma_k(\theta^{-1}(i)))}) \to C_k$.

To complete the proof, we need to show that if $\theta(\sigma_k(\theta^{-1}(i))) = \sigma_k(i)$ for all $1 \leqslant i \leqslant n$ then the permutation θ defines a colour-preserving automorphism of the graph $C_S(G)$.

We have already shown that θ must respect colours. It means that there is no directed edge (i, j) in $C_S(G)$ of colour c_k such that $(\theta(i), \theta(j))$ is an edge of another colour $c_{k'}$.

Thus, if θ does not define a colour-preserving automorphism of $C_S(G)$, then it does not define an automorphism at all. In other words, there is a colour c_k and an edge (i_0, j_0) of this colour such that $(\theta(i_0), \theta(j_0))$ is not an edge, i.e., θ does not respect the adjacence relation of $C_S(G)$.

By definition of $C_S(G)$, the pair (i_0, j_0) may be written as $(i_0, \sigma_k(i_0))$ and its image as $(\theta(i_0), \theta(\sigma_k(i_0)))$. Let $i_0 = \theta^{-1}(i)$, then $(i, \theta(\sigma_k(\theta^{-1}(i))))$ is not an edge of $C_S(G)$ while $(i, \sigma_k(i))$ is. However, as we established above, if M is an automorphism of $T_S(G)$, then the equality $\sigma_k(i) = \theta(\sigma_k(\theta^{-1}(i)))$ holds for all $1 \leqslant i \leqslant n$. $\diamond$ □

5.2. *Dependent products*

The construction described above for the second-order types may be applied with a slight modification to the dependent products.

The group G and the graph $C_S(G)$ are as above.

Recall that $K \to K'$ in LF means $(x : K)K'$, where $x \notin \mathrm{FV}(K')$ (it is not really a *dependent* product).

First, we define the kinds

$$C_1 \equiv \mathrm{Type} \to \mathrm{Type},\ C_2 \equiv C_1 \to \mathrm{Type},\ \ldots,\ C_m \equiv C_{m-1} \to \mathrm{Type}.$$

They are well formed in empty context and represent colours.

Then we take the variables $X_{1\div n} : \mathrm{Type}$ that will represent the elements of G and define the kinds $K_{ik} \equiv (\mathrm{El}(X_i) \to \mathrm{El}(X_{\sigma_k(i)})) \to C_k$.

That is, $X_i : \mathrm{Type}, X_{\sigma_k(i)} : \mathrm{Type} \vdash K_{ik}$ **kind**.

Instead of $T_S(G)$, the kind $K_S(G)$ (well formed in empty context) may be considered:

$$(X_{1\div n} : \mathrm{Type})(K_{11} \to \ldots K_{nm} \to \mathrm{Type}).$$

Theorem 5.6. The group of automorphisms of the kind $K_S(G)$ is isomorphic to the group of automorphisms of the graph $C_S(G)$ and thus isomorphic to G.

Its proof follows the same schema as the proof of Theorem 5.5. Instead of Lemmas 3.14, 3.24 is used repeatedly. The formulation (and proof) of the latter is much more complex,

but only simple instances are really needed because in $K_S(G)$ there is only one 'level of dependency' over $X_{1 \div n}$, and these variables are all of kind Type. (As a consequence, there will be no substitution of non-trivial isomorphism terms into kinds.)

6. Conclusion

To the author's knowledge, this paper is the first study of automorphisms of types. Theorems 5.5 and 5.6 show that automorphisms of types in higher order and dependent type systems are sufficient to represent arbitrary finite groups. Automorphisms in simply typed calculi $\lambda^1\beta\eta\pi*$ and $\lambda^1\beta\eta$ represent the groups of automorphisms of finite trees (Theorem 4.12). The fact that the presence of surjective pairing and terminal object does not change the class of representable groups is to some extent a surprise.

Automorphisms of types may be seen as a special kind of isomorphisms. Di Cosmo in his book (Di Cosmo 1995) (chapter 7 on related works and future perspectives) considered as principal applications of isomorphisms of types the library searches based on types as keys, equational matching and unification of types, dynamic composition of software components (based on matches performed modulo isomorphism), representation optimization. Since 1995, when the book was published, the studies of isomorphisms advanced essentially along these lines, see, e.g., Barthe (2005), Clairambault (2012), Delahaye (1999), Fiore (2004) and Fiore et al. (2006).

The groupoid structure described in Section 3.5 points to the applications related to coercive subtyping (Soloviev and Luo 2002; Luo et al. 2013).

Automorphisms do not change types, but change their elements, and this points towards applications of another kind. It seems that they may be especially of interest in a cryptographic context.

From a theoretical point of view, the main interest of automorphisms of types seems to be in the fact that they are able to represent complex algebraic structures, such as arbitrary finite groups. We already mentioned that it may be significant for foundational programs, that use isomorphism of types.

As to the future work, an obvious direction would be to extend this study to other systems of type theory. For example, it would be of interest to study the properties of automorphisms in the extensions of the systems considered above, such as the system with empty (or intersection) and sum types.

Theorems 4.2 and 4.12 permit to compute the group $\mathrm{Aut}(A)$ algorithmically. It would be of interest to obtain similar algorithmic description for any type A in $\lambda^2\beta\eta$, $\lambda^2\beta\eta\pi*$ or kind K in LF. We would like to investigate the possibilities to represent infinite groups in type theory. Representation of other algebraic structures (semigroups, monoids, rings and modules) may also be in the agenda.

The author would like to express his thanks to the organizers of the Trimester on Semantics of Proofs and Certified Mathematics at the Institite Henri Poincaré (Paris, April–July 2014) who partly supported his participation in this research program, in

particular to P.-L. Curien and H. Herbelin who later invited him to present an early version of the results of this paper at University Paris-7 (PPS seminar, March 2015). Thanks also to anonymous referees for their valuable remarks. This work was partially supported at IRIT by the CLIMT project, ANR-11-BS02-016-02, and by the Goverment of the Russian Federation Grant 074-U01 at ITMO University.

References

Asperti, A. and Longo, G. (1991). *Categories, Types and Structures. Category Theory for the Working Computer Scientist,* M.I.T. Press.

Babai, L. (1995). Automorphism groups, isomorphism, reconstruction. In: R. L. Graham et al., eds. *Handbook of Combinatorics,* Elsevier, vol. 2, 1447–1541.

Balat, V. and Di Cosmo, R. (1999). *A Linear Logical View of Linear Type Isomorphisms.* Lecture Notes in Computer Science, Springer-Verlag, vol. 1683, 250–265.

Barendregt, H. (1984). *The Lambda Calculus: Its Syntax and Semantics (revised edition),* North-Holland Plc.

Barthe, G. (2005). A computational view of implicit coercions in type theory. *Mathematical Structures in Computer Science* **15** (5) 839–874.

Brown, R., Higgins, Ph. G., and Sivera, R. (2011). *Nonabelian Algebraic Topology,* European Math. Society.

Bruce, K. and Longo, G. (1985). Provable isomorphisms and domain equations in models of typed languages. In: *Proceedings of the ACM Symposium on Theory of Computing (STOC 85)* 263–272.

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

Clairambault, P. (2012). Isomorphisms of types in the presence of higher-order references. *Logical Methods in Computer Science* **8** 1–28.

Coppo, M., Dezani-Ciancaglini, M., Margaria, I., Zacchi, M. (2017). Isomorphism of intersection and union types. *Mathematical Structures in Computer Science* **27** (5) 603–625.

Coppo, M., Dezani-Ciancaglini, M., Díaz-Caro, A., Margaria, I., Zacchi, M. (2016). Retractions in intersection types. In: N. Kobayashi (ed.) *ITRS 2016, EPTCS 242*, 31–47.

Coquand, T. and Huet, G. (1988). The calculus of constructions. *Information and Computation* **76** (23) 95–120.

Delahaye, D. (1999). Information retrieval in a coq proof library using type isomorphisms. In: T. Altenkirch et al., (eds.) *TYPES 1999*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1956, 131–147.

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

Di Cosmo, R. (1995). *Isomorphisms of Types: From Lambda-Calculus to Information Retrieval and Language Design,* Birkhauser.

Dosen, K. and Petric, Z. (1997). Isomorphic objects in symmetric monoidal closed categories. *Mathematical Structures in Computer Science* **7** 639–662.

Fiore, M. (2004). Isomorphisms of generic recursive polynomial types. In: *POPL '04*, New York, NY, USA, ACM, 77–88.

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) 35–50.

Frucht, R. (1938). Herstellung von Graphen mit vorgegebener abstrakten Gruppe. *Kompositio Math.* **6** 39–50.

Frucht, R. (1949). Graphs of degree three with a given abstract group. *Canadian Journal of Mathematics* **1** 365–378.

Goguen, H. (1994). A Typed Operational Semantics for Type Theory. PhD thesis, University of Edinburgh, UK.

Hankin, Chr. (1994). *Lambda Calculi: A Guide for Computer Scientists,* Clarendon Press, Oxford.

Harari, F. (1969). *Graph Theory,* Addison-Wesley PLC.

Hall, M. (Jr.) (1959). *The Theory of Groups,* The Macmillan Company, N.-Y.

Lambek, J. and Scott, P.J. (1988). *Introduction to Higher-Order Categorical Logic,* Cambridge Studies in Advanced Mathematics, Cambridge University Press, UK.

Longo, G., Milsted, K., and Soloviev, S. (1993). The genericity theorem and effective parametricity in polymorphic lambda-calculus. *Theoretical Computer Science* **121** 323–349.

Luo, Z. (1994). *Computation and Reasoning. A Type Theory for Computer Science*. International Series of Monographs on Computer Science, vol. **11**, Oxford Science Publications, Clarendon Press, Oxford, UK.

Luo, Z., Soloviev, S., and Xue, T. (2013). Coercive subtyping: Theory and implementation. *Information and Computation* **223** 18–42.

Mac Lane, S. (1976). Topology and logic as a source of algebra. *Bulletin of the American Mathematical Society* **82** (1) 1–40.

Marie-Magdeleine, L. (2009). Sous-typage coercitif en présence de réductions non-standards dans un système aux types dépendants. Thèse de doctorat. Université Toulouse-3 Paul Sabatier.

Pólya, G. (1937). Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica* **68** 145–254.

Soloviev, S. and Luo, Z. (2002). Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic* **113** (1–3) 297–322.

Solov'ev, S. V. (1983). The category of finite sets and cartesian closed categories. *Journal of Soviet Mathematics* **22** (3) 1387–1400.

Soloviev, S. V.(1993). A complete axiom system for isomorphism of types in closed categories. In: A. Voronkov, (ed.) *LPAR'93*, Lecture Notes in Artificial Intelligence, Springer-Verlag, vol. 698, 360–371.

Soloviev, S. (2015). On isomorphism of dependent products in a typed logical framework. In: *Post-proceedings of 20th International Conference on Types for Proofs and Programs, TYPES 2014,* May 12–15, 2014, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Paris, France, LIPICS, vol. 39, 275–288.

Stirling, C. (2013). Proof systems for retracts in simply typed lambda calculus. In: F. V. Fomin et al., (eds.) *ICALP (2)* Lecture Notes in Computer Science 7966, Springer-Verlag, 398–409.

The Univalent Foundations Program (2013). Homotopy Type Theory: Univalent Foundations of Mathematics. Available at `https://homotopytypetheory.org/book`, Institute for Advanced Study, Princeton.

White, A. T. (1984). *Graphs, Groups and Surfaces,* North-Holland, Amsterdam.