



**digital**

PARIS RESEARCH LABORATORY

## The Genericity Theorem and the Notion of Parametricity in the Polymorphic $\lambda$ -calculus

---

December 1992

Giuseppe Longo  
Kathleen Milsted  
Sergei Soloviev



---

## **The Genericity Theorem and the Notion of Parametricity in the Polymorphic $\lambda$ -calculus**

---

Giuseppe Longo  
Kathleen Milsted  
Sergei Soloviev

---

December 1992

---

## Publication Notes

This work will be published in a special issue of *Theoretical Computer Science* on Lambda Calculus, in honor of Corrado Böhm's 70th birthday. An extended abstract of this work also appears in the Proceedings of the 8th Annual IEEE Symposium on *Logic in Computer Science*, Montreal, Canada (June 20-23, 1993).

For further information, please contact Giuseppe Longo at LIENS(CNRS)-DMI, Ecole Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France. E-mail: [longo@dm.ens.fr](mailto:longo@dm.ens.fr)

© Digital Equipment Corporation and Ecole Normale Supérieure 1992

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for non-profit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by joint permission of the Paris Research Laboratory of Digital Equipment Centre Technique Europe (Rueil-Malmaison, France) and of the Laboratoire d'Informatique of the Ecole Normale Supérieure (Paris, France); an acknowledgement of the authors and individual contributors to the work; and all applicable portions of the copyright notice. All rights reserved.

## Abstract

In the polymorphic  $\lambda$ -calculus, one may explicitly define functions that take a type as input and return a term as output. This work focuses on how such functions depend on their input types. Indeed, these functions are generally understood to have an essentially constant meaning on input types. We show how the proof theory of the polymorphic  $\lambda$ -calculus suggests a clear syntactic description of this phenomenon. Namely, under a reasonable condition, we show that if two polymorphic functions agree on an input type, then they are, in fact, the same function. Equivalently, types are *generic* inputs to polymorphic functions.

## Résumé

Dans le  $\lambda$ -calcul polymorphe, on peut explicitement définir des fonctions qui prennent un type comme argument et qui renvoient un terme comme résultat. Le but de ce travail est de mieux comprendre la dépendance de ces fonctions vis-à-vis de leurs arguments types. En effet, ces fonctions sont généralement considérées comme étant essentiellement constantes par rapport aux arguments types. Nous montrons que la théorie syntaxique du  $\lambda$ -calcul polymorphe suggère une description claire de ce phénomène : sous une condition raisonnable, si deux fonctions polymorphes s'accordent sur un seul type, elles sont identiques. Autrement dit, les types sont des arguments *génériques* aux fonctions polymorphes.

## Keywords

Type theory; second-order lambda calculus; system F; parametric polymorphism; parametricity; genericity.

## Acknowledgements

We are greatly indebted to Pierre-Louis Curien who pointed out a fundamental error in a preliminary version of this work, as well as the connection to Reynolds's conditions. Thanks also to Jean Gallier for many passionate discussions about system F, to Eugenio Moggi and Roberto Di Cosmo for helpful comments and some early discussions on the Genericity Theorem, and to Simone Martini and the referees for valuable suggestions about this paper.

Giuseppe Longo's work was partially supported by a collaboration at Digital PRL. Sergei Soloviev's work was carried out at LIENS under a grant from the French Ministry for Research and Technology.

## Contents

1	Introduction	1
2	System F	3
3	System Fc	5
4	Roadmap to the Proof of Genericity	6
5	Type and Term Generalizers	8
6	Weak Genericity of F-equality	14
7	Quasi-Genericity of C*-equality	15
8	Commutativity of C*-equality with Reduction	17
9	The Genericity Theorem	21
10	Models	23
	References	26





## 1 Introduction

The use of types as explicit parameters, or variable types, is at the core of polymorphic (functional) languages, and was introduced, in Logic, by Girard [Gir71] and, in Computer Science, by Reynolds [Rey74]. The idea is that one may define formal functions that explicitly depend on input types. In  $\lambda$ -calculus notation, where capital  $X, Y, \dots$  stand for type variables, one may construct terms such as  $\lambda X.M$  which may be fed a type as input and give a term as output (in Logic jargon,  $\lambda X.M$  is a second-order term in impredicative Type Theory).

Originating with remarks by Strachey [Str67], a distinction was introduced on how these explicitly polymorphic functions should behave. Indeed, in computing, programs may depend on types. Overloaded functions, for example, may call different code according to the input type (or to the type of the input):  $+$  uses different code according to whether the addition is performed on (the type of) reals or integers, say. This sort of dependency of terms on types, known as *ad hoc* polymorphism, is an expressive feature of some programming languages, in particular when handled at run-time, and may suggest interesting and general formal systems (see [CGL92], say).

According to Strachey (and Reynolds) then, “proper” polymorphism, as opposed to the *ad hoc* variety, is the property that second-order terms have a *uniform* dependency on input types, or that their output terms do not “essentially” depend on input types. Note, though, that the output terms of, say,  $\lambda X.M$  applied to types  $\sigma$  and  $\tau$ , i.e.,  $(\lambda X.M)\sigma$  and  $(\lambda X.M)\tau$ , need not live in the same type. The point then is to understand how core systems, such as Girard-Reynolds system F [Gir71, Rey74] (also known as second-order  $\lambda$ -calculus), realize this uniform dependency property, known as *parametricity*, and compare terms possibly living in different types; more generally, to understand the functional behavior of formal functions such as  $\lambda X.M$ .

A semantic criterion for parametricity was proposed by Reynolds [Rey83, MR91] as an invariance property under relations between type values. In short, if a relation is given on type parameters  $\sigma$  and  $\tau$ , then (the interpretation of)  $\lambda X.M$ , applied to (the meaning of)  $\sigma$  and  $\tau$ , should send related elements of  $\sigma$  and  $\tau$  to related elements in the types of the outputs. This is known as *relational parametricity*, and a syntactic treatment of it is given in [ACC93] and in [PA93].

Another approach to parametricity was proposed by Bainbridge et al. [BFSS90]. Consider  $\lambda x : X.N$ . Is it the case that  $\lambda x : X.N$  depends naturally on  $X$ , in the sense of natural transformations of Category Theory? Indeed, natural transformations are the core means of expressing uniformity on objects (as interpretation of types) in categories. Unfortunately, natural transformations act on functors, whereas, in general categories, variable types are not functors. The counterexample is straightforward: the map from  $X$  to  $X \rightarrow X$  (the arrow type) should be at once a covariant and contravariant functor. A partial solution, in the context of the typed  $\lambda$ -calculus, may be given by considering categories where maps are only retractions (as in [Sco72, SP82, Gir86]) or isomorphisms (as in [DL89]). This is fine for specific purposes, as in those papers, but does not describe the situation in the full generality

of a model theoretic approach. On the other hand, this issue of contra/covariant functors was partly at the origin of relevant generalizations of the notion of functor in mathematics, for example [EK66]; see also [Mac71]. In this line of work, Bainbridge et al. propose to interpret terms as dinatural transformations, yet another elegant categorical notion derived from tensor algebra and algebraic topology. The rub is that, in general, dinatural transformations do not compose, while terms do; however, the interpretation works well (i.e., it is compositional) on relevant models (see [BFSS90, FGSS88, GSS]), in particular on models of relational parametricity as formalized in [PA93]. On essentially similar lines, Freyd suggested a novel notion of structor in order to understand, categorically, the notion of uniformity inherent in second-order  $\lambda$ -terms.

These attempts suggested brand new constructions and relevant mathematics, but seem still insufficient to fill the essential gap between the parametricity of second-order  $\lambda$ -calculus and the uniformity with respect to objects (and functors) as expressed by natural transformations in Category Theory. This is probably one of the few mismatches (together with subtyping versus subobjects) out of many deep connections between types and objects, terms and morphisms, as summarized, say, in [AL91] and [LS86]. A survey and a classification of the various forms of parametricity is proposed in [Lon93].

In this paper, we consider a weak extension of system F, suggested by the following simple result of Girard in [Gir71]: given a type  $\sigma$ , if one takes a term  $J_\sigma$  such that, for any type  $\tau$ ,  $J_\sigma\tau$  reduces to 1 if  $\sigma = \tau$ , and reduces to 0 if  $\sigma \neq \tau$ , then  $F+J_\sigma$  does not normalize. Since system F normalizes,  $J_\sigma$  is not definable in F. The point here is that the polymorphic term  $J_\sigma$  gives essentially different output terms, which live in the same type, according to the (values of the) input types. Then, a first point in our understanding of parametricity is that a polymorphic term that gives outputs in the same type for all input types, must be constant. This is expressed by the following equational scheme:

$$\text{(Axiom C)} \quad M\tau = M\tau' \quad \text{for } \Gamma \vdash M : \forall X.\sigma \text{ and } X \notin FV(\sigma)$$

That is, if the outputs of a polymorphic term  $M$ , applied to any type, all live in the same type, then these outputs are simply equal. Axiom C is not provable in F, but it is compatible with F, that is, system F may be consistently extended with it. Indeed, a generalization of Axiom C appears in the system  $F_{<}$ : [CMMS91] which extends system F with subtyping; see rule *Eq appl2*. In our view, the compatibility of Axiom C with system F is one thing to be noted in order to understand parametricity. Moreover, all models that yield the dinatural interpretation of terms in [BFSS90] realize Axiom C, as do PER models in realizability topoi and Girard's models over dI-domains and stable maps. From [ACC93] and [Has93], it also turns out that Axiom C is realized by all models that satisfy Reynolds's relational parametricity condition [MR91]. A categorical characterization of models realizing Axiom C will be outlined in Section 10.

Consider now  $F_c$ , the extension of system F with Axiom C. The main result of this paper is the following theorem:

**(Genericity Theorem)**    Assume  $M$  and  $N$  live in the same type  $\forall X.\sigma$   
                                  If  $M\tau =_{Fc} N\tau$  for some type  $\tau$ , then  $M =_{Fc} N$

The reader should notice where intended parentheses and existential quantification are located, and also, that there is no restriction on  $\sigma$ . The Genericity Theorem states the rather strong fact that, in  $F_c$ , if two second-order terms coincide on an input type, then they are, in fact, the same function. Or, equivalently, that each input type acts as a *generic* input, as a variable. It also says, in a sense, that there are “very few” polymorphic functions. Note that the Genericity Theorem does not hold in  $F$ . Take, for example,  $x : \forall X.\sigma$  with  $X \notin FV(\sigma)$ , and consider  $M \equiv \lambda X.x\tau$  and  $N \equiv \lambda X.xX$ , both of type  $\forall X.\sigma$ . Then,  $M\tau =_F N\tau$  but  $M$  and  $N$  are not  $F$ -equal. Indeed, as pointed out by Furio Honsell and one of the referees, it is easy to show that  $F_c$  is the least equational extension of  $F$  which yields the Genericity Theorem.

Observe finally that, although all models of relational parametricity realize Axiom C, it may be shown that no such model realizes Genericity as an implication. This is a delicate issue, hinted at in Section 10 and discussed extensively in [Lon93]. In the following sections, we recall system  $F$  and introduce our syntactic conventions, describe system  $F_c$ , and prove the Genericity Theorem.

## 2 System F

The language of system  $F$  consists of *types* and *terms*. A type is either a type variable, a function type, or a polymorphic type, while a term is either a variable, an abstraction, an application, a type abstraction, or a type application. Types and terms have the following syntax:

$$\begin{array}{ll} \text{Types} & \sigma ::= X \mid \sigma \rightarrow \tau \mid \forall X.\sigma \\ \text{Terms} & M ::= x \mid \lambda x:\sigma.M \mid MN \mid \lambda X.M \mid M\tau \end{array}$$

We will use  $\sigma, \tau, \rho, \mu, \nu$  for types and  $M, N$  for terms, while for variables, we will use  $X, Y, Z$  for type variables and  $x, y, z$  for term variables. Following the usual conventions for minimizing parentheses, applications associate to the left,  $\rightarrow$  associates to the right, and the scope of  $\forall$  and  $\lambda$  extends as far to the right as possible. For any type or term  $P$ , the set of its free (type and term) variables is defined as usual, and written  $FV(P)$ . Capture-avoiding type substitution and term substitution is also defined as usual on types and terms, and written  $[\tau/X]P$  and  $[M/x]P$ , respectively.

Assignment of types to terms takes place relative to a set of *variable declarations*, where each declaration assigns a unique type to a term variable. We will use  $\Gamma$  for a set of declarations, and we write  $\Gamma, x : \sigma$  to extend  $\Gamma$  with a new declaration  $x : \sigma$ , where  $x$  must not occur in  $\Gamma$ . The substitution of a type in a set of declarations,  $[\tau/X]\Gamma$ , is defined component-wise as substitution into the type of each declaration in  $\Gamma$ .

A *type assignment* is a meta-expression of the form  $\Gamma \vdash M : \sigma$ , which asserts that term  $M$  has, or lives in, type  $\sigma$ , relative to the declarations in  $\Gamma$ . The following rules define valid type assignments.

### Type Assignment Rules

$$\begin{array}{ll}
 \text{(declaration)} & \Gamma, x : \sigma \vdash x : \sigma \\
 \\
 (\rightarrow\text{-intro}) & \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma. M : \sigma \rightarrow \tau} \quad (\rightarrow\text{-elim}) \quad \frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau} \\
 \\
 * (\forall\text{-intro}) & \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \lambda X. M : \forall X. \sigma} \quad (\forall\text{-elim}) \quad \frac{\Gamma \vdash M : \forall X. \sigma}{\Gamma \vdash M\tau : [\tau/X]\sigma} \\
 & * \text{ for } X \text{ not free in the type of} \\
 & \text{any free term variable in } M
 \end{array}$$

Note the restriction on the  $\forall$ -intro rule: without it, it would be possible to prove inconsistencies such as  $x : Y \vdash x : Z$ . This restriction will show up frequently later.

Equality of terms is defined by the following schemes and rules:

### Equational Schemes and Rules

$$\begin{array}{ll}
 (\beta_1) \quad (\lambda x : \sigma. M)N = [N/x]M & (\beta_2) \quad (\lambda X. M)\tau = [\tau/X]M \\
 (\eta_1) \quad \lambda x : \sigma. Mx = M \text{ for } x \notin FV(M) & (\eta_2) \quad \lambda X. MX = M \text{ for } X \notin FV(M) \\
 (\xi_1) \quad \frac{M = N}{\lambda x : \sigma. M = \lambda x : \sigma. N} & (\xi_2) \quad \frac{M = N}{\lambda X. M = \lambda X. N} \\
 (\text{app}_1) \quad \frac{M_1 = M_2 \quad N_1 = N_2}{M_1 N_1 = M_2 N_2} & (\text{app}_2) \quad \frac{M = N}{M\tau = N\tau} \\
 (\text{refl}) \quad M = M & (\text{sym}) \quad \frac{M_1 = M_2}{M_2 = M_1} \quad (\text{trans}) \quad \frac{M_1 = M_2 \quad M_2 = M_3}{M_1 = M_3}
 \end{array}$$

We will use the symbol  $\equiv$  for syntactic identity. For types,  $\sigma = \tau$  is the same as  $\sigma \equiv \tau$  while, for terms,  $M \equiv N$  implies  $M = N$  but not vice-versa.

Reduction of terms is defined as usual by the closure of the following rules:

$$\begin{array}{ll}
 (\beta_1) \quad (\lambda x : \sigma. M)N \longrightarrow_{\beta_1} [N/x]M & (\beta_2) \quad (\lambda X. M)\tau \longrightarrow_{\beta_2} [\tau/X]M \\
 (\eta_1) \quad \lambda x : \sigma. Mx \longrightarrow_{\eta_1} M \text{ for } x \notin FV(M) & (\eta_2) \quad \lambda X. MX \longrightarrow_{\eta_2} M \text{ for } X \notin FV(M)
 \end{array}$$

We will write  $\longrightarrow_F$  for the union of these reductions.

The following important properties hold for system F.

### Unique Typing

*A well-typed term lives in a unique type: if  $\Gamma \vdash M : \sigma$  and  $\Gamma \vdash M : \tau$  then  $\sigma = \tau$ .*

### Strong Normalization

*There are no infinite reduction sequences from well-typed terms.*

### Church-Rosser

*If  $M \rightarrow_F M_1$  and  $M \rightarrow_F M_2$  then there exists an  $M_0$  such that  $M_1 \rightarrow_F M_0$  and  $M_2 \rightarrow_F M_0$ .*

### Equational Church-Rosser

*If  $M_1 = M_2$  then there exists an  $M_0$  such that  $M_1 \rightarrow_F M_0$  and  $M_2 \rightarrow_F M_0$ .*

## 3 System Fc

System Fc is formed by adding the following equational scheme to system F:

$$(\textbf{Axiom C}) \quad M\tau = M\tau' \quad \text{for } \Gamma \vdash M : \forall X.\sigma \text{ and } X \notin FV(\sigma)$$

That is, if the outputs of polymorphic function  $M$  live in a type  $\sigma$  that does not depend on  $M$ 's input type, then the outputs are equal, regardless of the input type. Or, equivalently,  $M$  is constant.

Axiom C equates more terms than in system F. We will write  $M =_F N$  for F-equations, and  $M =_{Fc} N$  for Fc-equations. Clearly, Axiom C is not provable in system F. Take  $x : \forall X.\sigma$  with  $X \notin FV(\sigma)$ , and apply Axiom C to  $x$ . This gives

$$x\tau =_{Fc} x\rho$$

These two terms would be equated in system F only if  $\tau = \rho$ .

Since system Fc adds no new terms, types, typing rules, or reductions, it enjoys the same *non-equational* properties as system F, such as unique typing of terms, as well as strong normalization and the Church-Rosser property (relative to  $\rightarrow_F$ ). However, a number of *equational* properties fail for Fc, in particular, the equational Church-Rosser property: for example, even though  $x\tau =_{Fc} x\rho$  above, there is no common term to which both  $x\tau$  and  $x\rho$  reduce.

In the proof of the Genericity Theorem, it will generally be more convenient to use a term with a type substitution structure such as  $[\tau/X]M$  instead of a polymorphic application  $M\tau$ . Thus, we may use the following formulation of Axiom C:

$$(\textbf{Axiom C}^*) \quad [\tau/X]M = [\tau'/X]M \quad \text{for } \Gamma \vdash M : \sigma \text{ and } X \notin FV(\Gamma) \cup FV(\sigma)$$

It is simple to prove that Axiom C and Axiom C\* are equivalent. We give the proof to stress the extra side-condition  $X \notin FV(\Gamma)$  on Axiom C\* and its relation to the side-condition on

$\forall$ -introduction. These conditions will appear frequently in the later proofs. We will write  $M =_c N$  and  $M =_{c^*} N$  if  $M$  and  $N$  are equal by only applications of Axiom C and Axiom C\* respectively.

**Remark:** *Axiom C\* is equivalent to Axiom C.*

*Axiom C implies Axiom C\*:*

Assume that  $\Gamma \vdash M : \sigma$  and  $X \notin FV(\Gamma) \cup FV(\sigma)$ .

Since  $X \notin FV(\Gamma)$ , then  $X$  is not free in the type of any free term variable in  $M$ .

So, by  $\forall$ -intro,  $\Gamma \vdash \lambda X.M : \forall X.\sigma$ . Also,  $X \notin FV(\sigma)$ .

Thus, by Axiom C and  $\beta_2$ ,  $[\tau/X]M =_{\beta_2} (\lambda X.M)\tau =_c (\lambda X.M)\tau' =_{\beta_2} [\tau'/X]M$ .

*Axiom C\* implies Axiom C:*

Assume that  $\Gamma \vdash M : \forall X.\sigma$  and  $X \notin FV(\sigma)$ .

Let  $Z$  be a fresh variable. Then,  $\Gamma \vdash MZ : \sigma$  and  $Z$  is not free in any of  $\Gamma, M, \sigma$ .

Thus, by Axiom C\*,  $M\tau \equiv [\tau/Z](MZ) =_{c^*} [\tau'/Z](MZ) \equiv M\tau'$ . ■

## 4 Roadmap to the Proof of Genericity

In this section, we outline the route to the proof of the Genericity Theorem:

*Assume  $M$  and  $N$  live in the same type  $\forall X.\sigma$   
If  $M\tau =_{Fc} N\tau$  for some type  $\tau$ , then  $M =_{Fc} N$*

The hard part is to prove the following Main Lemma, which is a substitution formulation of the Theorem:

*Assume  $M$  and  $N$  live in the same type  $\sigma$   
If  $[\tau/X]M =_{Fc} [\tau/X]N$  for some type  $\tau$ , then  $M =_{Fc} N$*

The first remark to be made about the proof is that it is not an induction. The point is that corresponding subterms of Fc-equal terms do not need to live in the same type. The following example illustrates why.

**Example:** Assume  $x : \forall Y.Y$  and  $z : \forall Y_1.\forall Y_2.Y_1 \rightarrow Y_2$ .

Let  $X$  and  $Z$  be fresh type variables.

Then, Axiom C\* can be applied to the term  $zZX(xZ) : X$  to obtain

$$z\tau X(x\tau) =_{Fc} z\rho X(x\rho)$$

Note that subterms  $z\tau X$  and  $z\rho X$  live in different types.

However, this example also provides a hint to the proof of Genericity. Observe that the Fc-equality  $z\tau X(x\tau) =_{Fc} z\rho X(x\rho)$  is obtained via the intermediate term  $zZX(xZ)$  to

which Axiom  $C^*$  is applied. Furthermore,  $z\tau X(x\tau)$  and  $z\rho X(x\rho)$  are both instances of this term, using type substitutions  $[\tau/Z]$  and  $[\rho/Z]$  respectively. Approximately then, the hint is this: given two Fc-equal terms, construct a common term that can be instantiated to the two terms by type substitutions, and to which Axiom  $C^*$  can be applied.

The proof thus begins in Section 5 by developing the notion of a *generalizer* for second-order terms. This is a novel idea for the polymorphic  $\lambda$ -calculus, although it is, of course, related to generalizers and anti-unifiers of first-order calculi. Given two second-order terms that are identified by type substitutions, we construct a common term that can be instantiated, by type substitutions, to the original terms. Similarly, we can construct a common type that can be instantiated, by type substitutions, to two given types. Furthermore, if the two terms live in two different types, then the generalizer of the terms lives in the generalizer of the types. Note that this notion of generalizer uses *type* substitutions, not term substitutions (as is usual for first-order terms).

In Section 6, we use generalizers to prove the following *Weak Genericity* theorem:

*Assume  $M$  and  $N$  live in the same type  $\sigma$*   
*If  $[\tau/X]M =_F [\tau/X]N$  for some type  $\tau$ , then  $M =_{Fc} N$*

The weakness arises because an F-equality is used in the premise instead of an Fc-equality. This theorem is used in the final result, and it marks an important halfway-point in the overall proof.

The proof proceeds next with a property of  $C^*$ -equality that we call *Quasi-Genericity*: if a term has a type substitution structure (is of the form  $[\tau/X]M$ ) and Axiom  $C^*$  is applied to it, then that exact type substitution structure is preserved, that is, the result is of the form  $[\tau/X]N$ , and, moreover,  $M =_{c^*} N$ . The proof of this also uses generalizers and is given in Section 7, where we also give a counter-example to show that F-equality does **not** satisfy this property. Using Quasi-Genericity, we are able to prove another weak version of Genericity, this time with  $C^*$ -equality in the premise instead of Fc-equality:

*Assume  $M$  and  $N$  live in the same type  $\sigma$*   
*If  $[\tau/X]M =_{c^*} [\tau/X]N$  for some type  $\tau$ , then  $M =_{Fc} N$*

Finally, in Section 9, we draw all the pieces together to prove the Main Lemma. This involves examining the chain of F and  $C^*$ -equalities  $[\tau/X]M =_{Fc} [\tau/X]N$ . Unfortunately, F-equality and  $C^*$ -equality do not commute, but, in Section 8, we show that forward  $\beta_1\beta_2\eta_1$  reduction (but not  $\eta_2$  reduction) commutes with  $C^*$ -equality. Using this fact, the Church-Rosser property for F-reductions, and Quasi-Genericity of  $C^*$ -equality, we “push” the  $[\tau/X]$  substitution structure from  $[\tau/X]M$  through the chain so that each node in the chain has the form  $[\tau/X]M_i$  for some  $M_i$  with  $M =_{Fc} M_i$ . Finally, we use Weak Genericity of F and  $C^*$ -equality to show that the final node  $[\tau/X]N$  in the chain is such that  $M =_{Fc} N$ . This gives the Genericity result.

## 5 Type and Term Generalizers

In this section, we construct a notion of *generalizer* for types and terms. In short, a generalizer of two types (terms) may be instantiated, using type substitutions, to the two types (terms), under suitable conditions. Generalizers are used in later sections, where we show that, in the case of term generalizers, the typing of the generalizer permits Axiom  $C^*$  to be applied to it, resulting in Fc-equality of the two terms.

As motivation, consider two terms  $M_1$  and  $M_2$  such that  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ . Then, approximately, a generalizer of  $M_1$  and  $M_2$ , with respect to a fresh type variable  $Z$ , is a term  $M_0$  such that, for suitable types  $\mu_1, \mu_2$ :

$$\begin{aligned} [\mu_1/Z]M_0 &\equiv M_1 \\ [\mu_2/Z]M_0 &\equiv M_2 \end{aligned}$$

In other words, if two terms can be unified as above, then we construct a common “term schema” which can be instantiated, by type substitutions, to both of them. This is an abstract notion of a generalizer though, and the generalizers that we construct here require more details, including an analysis of occurrences of  $\tau$  in  $\rho$  or  $\rho$  in  $\tau$ .

**Definition:**  $in_k$

If there are  $k \geq 0$  occurrences of type  $\tau$  in type  $\rho$ , we will write  $\tau \text{ in}_k \rho$ .

**Definition:** Context

Let  $\tau, \rho, \rho'$  be types and let  $X$  be a type variable. We say that  $\rho'$  is an  $X$ -**context** for  $\tau$  in  $\rho$  if  $[\tau/X]\rho' = \rho$ .

If  $\tau \text{ in}_k \rho$  with  $k \geq 0$ , then, given fresh  $X$ , there are  $2^k$  different  $X$ -contexts for  $\tau$  in  $\rho$ . We will assume given an enumeration of these contexts, which we will write as  $\rho_1^X, \dots, \rho_h^X$  where  $h = 2^k$ . By convention, we take  $\rho_1^X$  to be  $\rho$ . For example, if  $\tau = \rho$ , then there are two  $X$ -contexts for  $\tau$  in  $\rho$ :  $\rho_1^X = \rho$  and  $\rho_2^X = X$ .

**Substitution Convention**

Let  $P_1, P_2$  be either two terms, or two types, or two sets of variable declarations.

If  $[\tau/X]P_1 \equiv [\rho/Y]P_2$  for some types  $\tau$  and  $\rho$ , then we will assume, with no loss of generality, that, by variable renaming,  $X$  and  $Y$  are not free in  $\tau$  and  $\rho$ .



**Definition: Generalizer**

Let  $P_1, P_2$  be either two terms, or two types, or two sets of variable declarations, such that  $[\tau/X]P_1 \equiv [\rho/Y]P_2$  for some types  $\tau$  and  $\rho$ .

- Case:  $\tau \text{ in}_k \rho$  for  $k > 0$ .

Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , we say that  $P_0$  is a  $Z_0, \dots, Z_h$ -**generalizer** of  $P_1$  and  $P_2$  iff  $X$  and  $Y$  are not free in  $P_0$  and

$$\begin{aligned} [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] P_0 &\equiv P_1 \\ [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] P_0 &\equiv P_2 \end{aligned}$$

where  $\rho_1^X, \dots, \rho_h^X$  are the  $X$ -contexts for  $\tau$  in  $\rho$ .

- Case:  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and the previous case does not apply.

Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , we say that  $P_0$  is a  $Z_0, \dots, Z_h$ -**generalizer** of  $P_1$  and  $P_2$  iff  $X$  and  $Y$  are not free in  $P_0$  and

$$\begin{aligned} [\rho/Z_0, X/Z_1, \dots, X/Z_h] P_0 &\equiv P_1 \\ [Y/Z_0, \tau_1^Y/Z_1, \dots, \tau_h^Y/Z_h] P_0 &\equiv P_2 \end{aligned}$$

where  $\tau_1^Y, \dots, \tau_h^Y$  are the  $Y$ -contexts for  $\rho$  in  $\tau$ .

Observe that, if  $\tau = \rho$ , then the first case of the definition applies, by  $\tau \text{ in}_1 \rho$ , giving

$$\begin{aligned} [X/Z_0, \rho/Z_1, X/Z_2] P_0 &\equiv P_1 \\ [\tau/Z_0, Y/Z_1, Y/Z_2] P_0 &\equiv P_2 \end{aligned}$$

If  $\tau$  and  $\rho$  are unrelated (i.e., they do not occur in each other), then the second case applies, by  $\rho \text{ in}_0 \tau$ :

$$\begin{aligned} [\rho/Z_0, X/Z_1] P_0 &\equiv P_1 \\ [Y/Z_0, \tau/Z_1] P_0 &\equiv P_2 \end{aligned}$$

Indeed, no matter how  $\tau$  and  $\rho$  are related, only one case of the definition applies: for example, one cannot have both  $\tau \text{ in}_0 \rho$  and  $\rho \text{ in}_0 \tau$ , nor both  $\rho \text{ in}_0 \tau$  and  $\tau \text{ in}_k \rho$ .

**Lemma 5.1 (Type Generalization)**

Let  $\sigma_1, \sigma_2$  be two types such that  $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$  for some types  $\tau$  and  $\rho$ . Assume that  $k$  is given either by  $\tau \text{ in}_k \rho$  for  $k > 0$ , or  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , there exists a type  $\sigma_0$  that is a  $Z_0, \dots, Z_h$ -generalizer of  $\sigma_1$  and  $\sigma_2$ .

Proof: Let  $\sigma = [\tau/X]\sigma_1 = [\rho/Y]\sigma_2$  and perform the following markings:

- Mark in  $\sigma$  those occurrences of  $\tau$  that derive from  $\sigma_1$  by a  $[\tau/X]$  substitution.
- Mark in  $\sigma$  those occurrences of  $\rho$  that derive from  $\sigma_2$  by a  $[\rho/Y]$  substitution.

Consider first the case where  $\tau \text{ in}_k \rho$  for  $k > 0$ .

Observe that some of the marked  $\tau$ s may appear in a marked  $\rho$ .

Construct then  $\sigma_0$  from  $\sigma$  by the following procedure:

1. Replace by  $Z_0$  all marked  $\tau$ s that do not occur in a marked  $\rho$ .
2. Consider now a marked  $\rho$ , possibly containing marked  $\tau$ s.  
 Let  $\rho_i^X$  be the corresponding  $X$ -context in  $\rho$  for the marked  $\tau$ s. (If there are no marked  $\tau$ s, this will be  $\rho_1^X \equiv \rho$ ). Replace the marked  $\rho$  by  $Z_i$ .  
 In the alternative case,  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case, observe that some of the marked  $\rho$ s may appear in a marked  $\tau$ . Then, apply the dual construction procedure, where the roles of  $\rho$  and  $\tau$  in steps 1 and 2 are interchanged, and  $\tau_i^Y$ , the  $Y$ -contexts for  $\rho$  in  $\tau$ , are used instead of  $\rho_i^X$ , the  $X$ -contexts for  $\tau$  in  $\rho$ . ■

In the following lemma, we show that, once fresh variables  $Z_0, \dots, Z_h$  are fixed, then the generalizer of two types is unique. This lemma makes explicit use of the substitution convention, i.e., that  $X, Y \notin FV(\tau) \cup FV(\rho)$ , without which it would fail.

### Lemma 5.2 (Uniqueness of Type Generalizer)

Let  $\sigma_1, \sigma_2$  be two types such that  $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$  for some types  $\tau$  and  $\rho$ . Assume that  $k$  is given either by  $\tau \text{ in}_k \rho$  for  $k > 0$ , or  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , the  $Z_0, \dots, Z_h$ -generalizer of  $\sigma_1$  and  $\sigma_2$  is unique.

Proof: Assume first that  $\tau \text{ in}_k \rho$  for  $k > 0$ .

Let  $\sigma_0$  and  $\sigma'_0$  be two  $Z_0, \dots, Z_h$ -generalizers of  $\sigma_1, \sigma_2$ . Then, by definition,

$$[X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma_0 = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0 \quad (1)$$

$$[\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma_0 = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0 \quad (2)$$

with  $X$  and  $Y$  not free in  $\sigma_0$  or  $\sigma'_0$ . We will show that  $\sigma_0 = \sigma'_0$  by induction on  $\sigma_0$ .

Subcase: Assume that  $\sigma_0 \equiv Z_0$ . Then, (1) and (2) become

$$X = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0$$

$$\tau = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0$$

We now consider the possible choices for  $\sigma'_0$ . Clearly,  $\sigma'_0$  cannot be  $X$  since  $X \notin FV(\sigma'_0)$ . Nor can  $\sigma'_0$  be  $\tau$  since then, (1) becomes  $X = \sigma_1 = \tau$  but, by the substitution convention,  $X \notin FV(\tau)$ . Further,  $\sigma'_0$  cannot be  $Z_i$  for some  $i = 1 \dots h$ , because then (2) becomes  $\tau = \sigma_2 = Y$  but, by the substitution convention again,  $Y \notin FV(\tau)$ . The only choice is  $\sigma'_0 \equiv Z_0 = \sigma_0$ .

Subcase: Assume that  $\sigma_0 \equiv Z_i$  for some  $i = 1 \dots h$ . Then, (1) and (2) become

$$\rho_i^X = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0$$

$$Y = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0$$

First,  $\sigma'_0$  cannot be  $Y$  since  $Y \notin FV(\sigma'_0)$ . Furthermore,  $\sigma'_0$  cannot be  $\rho_i^X$  since, for  $i = 1$ , (2) becomes  $Y = \sigma_2 = \rho_1^X = \rho$  but, by the substitution convention,  $Y \notin FV(\rho)$ , and, for  $i = 2 \dots h$ ,  $X \in FV(\rho_i^X)$  but  $X \notin FV(\sigma'_0)$ . Also,  $\sigma'_0$  cannot be  $Z_0$  since then, (2) becomes  $Y = \sigma_2 = \tau$  but, by the substitution convention again,  $Y \notin FV(\tau)$ . Similarly,  $\sigma'_0$  cannot be  $Z_j$  for some  $j = 1 \dots h$  and  $j \neq i$  since then, (1) becomes  $\rho_i^X = \sigma_1 = \rho_j^X$  but  $\rho_i^X \neq \rho_j^X$  for  $i \neq j$ . The only choice is  $\sigma'_0 \equiv Z_i = \sigma_0$ .

Subcase: Assume that  $\sigma_0 \equiv Z \neq Z_i$  for  $i = 0 \dots h$ . Then, (1) and (2) become

$$Z = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0$$

$$Z = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0$$

Since  $X$  and  $Y$  are not free in  $\sigma_0$ , then  $Z \neq X$  and  $Z \neq Y$  and, moreover,  $\sigma'_0$  cannot be  $Z_i$  for any  $i = 0 \dots h$ . The only choice is  $\sigma'_0 \equiv Z = \sigma_0$ .

Subcase: Assume that  $\sigma_0 \equiv \sigma \rightarrow \mu$ . Then, (1) and (2) become

$$[X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] (\sigma \rightarrow \mu) = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0$$

$$[\tau/Z_0, Y/Z_1, \dots, Y/Z_h] (\sigma \rightarrow \mu) = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0$$

Remark that  $\sigma'_0$  cannot be  $Z_i$  for any  $i = 0 \dots h$  since, then, a  $\rightarrow$  type would be on the left of (1) and (2) but a type variable would be on the right ( $X$  in (1) and  $Y$  in (2)). So,  $\sigma'_0$  must be of the form  $\sigma' \rightarrow \mu'$ , with  $\sigma, \sigma'$  and  $\mu, \mu'$  satisfying equations similar to (1) and (2). By induction,  $\sigma = \sigma'$  and  $\mu = \mu'$ . Hence,  $\sigma'_0 \equiv \sigma' \rightarrow \mu' = \sigma \rightarrow \mu = \sigma_0$ .

Subcase: Assume that  $\sigma_0 \equiv \forall Z. \sigma$ . Then, (1) and (2) become

$$[X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] (\forall Z. \sigma') = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] \sigma'_0$$

$$[\tau/Z_0, Y/Z_1, \dots, Y/Z_h] (\forall Z. \sigma') = \sigma_2 = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma'_0$$

As with the previous case,  $\sigma'_0$  cannot be  $Z_i$  for any  $i = 0 \dots h$ . So,  $\sigma'_0$  must be of the form  $\forall Z. \sigma'$ . By induction,  $\sigma = \sigma'$ . Hence,  $\sigma'_0 \equiv \forall Z. \sigma' = \forall Z. \sigma = \sigma_0$ .

Treat dually  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. ■

### Lemma 5.3

Let  $\sigma_1, \sigma_2, \mu_1, \mu_2$  be types such that  $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$  and  $[\tau/X]\mu_1 = [\rho/Y]\mu_2$ . Assume that  $k$  is given either by  $\tau \text{ in}_k \rho$  for  $k > 0$ , or  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , let  $\sigma_0$  and  $\mu_0$  be the  $Z_0, \dots, Z_h$ -generalizers of  $\sigma_1, \sigma_2$  and  $\mu_1, \mu_2$ , respectively. Then, for any  $Z$  different from  $Z_0, \dots, Z_h$ ,  $[\mu_0/Z]\sigma_0$  is the  $Z_0, \dots, Z_h$ -generalizer of  $[\mu_1/Z]\sigma_1$  and  $[\mu_2/Z]\sigma_2$ .

Proof: by expanding  $\sigma_1, \sigma_2$  and  $\mu_1, \mu_2$  in terms of their generalizers.

### Lemma 5.4 (Generalization of Declarations)

Let  $\Gamma_1, \Gamma_2$  be two sets of declarations such that  $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$ . Assume that  $k$  is given either by  $\tau \text{ in}_k \rho$  for  $k > 0$ , or  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , there exists a set of declarations  $\Gamma_0$  that is a unique  $Z_0, \dots, Z_h$ -generalizer of  $\Gamma_1$  and  $\Gamma_2$ .

Proof: Since  $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$ , then  $\Gamma_1$  and  $\Gamma_2$  must declare the same term variables.

Thus, we can assume that  $\Gamma_1 \equiv x_1 : \sigma_1^1, \dots, x_n : \sigma_n^1$  and  $\Gamma_2 \equiv x_1 : \sigma_1^2, \dots, x_n : \sigma_n^2$

with  $[\tau/X]\sigma_i^1 = [\rho/Y]\sigma_i^2$  for  $i = 1 \dots n$ .

Furthermore, by assumption on  $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$ , the substitution convention applies to each  $[\tau/X]\sigma_i^1 = [\rho/Y]\sigma_i^2$ .

So, for  $i = 1 \dots n$ , construct the unique  $Z_0, \dots, Z_h$ -generalizer  $\sigma_i^0$  of  $\sigma_i^1$  and  $\sigma_i^2$ .

Then,  $\Gamma_0 \equiv x_1 : \sigma_1^0, \dots, x_n : \sigma_n^0$  is the unique  $Z_0, \dots, Z_h$ -generalizer of  $\Gamma_1$  and  $\Gamma_2$ . ■

The next theorem is the main result of this section. It constructs a well-typed generalizer of two terms living in two *different* types. Uniqueness of type generalizers turns out to be essential in the proof (see the  $\rightarrow$ -elim case). The point to note is not just that we can construct a generalizer for  $M_1$  and  $M_2$ , but that we can construct one that is well-typed, and that lives in the type generalizer of the types of  $M_1$  and  $M_2$ .

### Theorem 5.5 (Term Generalization)

Let  $\Gamma_1 \vdash M_1 : \sigma_1$  and  $\Gamma_2 \vdash M_2 : \sigma_2$  be such that  $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$  and  $[\tau/X]M_1 \equiv [\rho/Y]M_2$  for some types  $\tau$  and  $\rho$ . Assume that  $k$  is given either by  $\tau \text{ in}_k \rho$  for  $k > 0$ , or  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and not the previous case. Let  $h = 2^k$ . Given fresh type variables  $Z_0, \dots, Z_h$ , there exist a set of declarations  $\Gamma_0$ , a term  $M_0$ , and a type  $\sigma_0$  that are unique  $Z_0, \dots, Z_h$ -generalizers of  $\Gamma_1, \Gamma_2$ ;  $M_1, M_2$ ; and  $\sigma_1, \sigma_2$ , respectively, and such that  $\Gamma_0 \vdash M_0 : \sigma_0$ .

Proof: Construct  $\Gamma_0, M_0, \sigma_0$  by induction on the derivation of  $\Gamma_1 \vdash M_1 : \sigma_1$ .

Observe first that  $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$  since  $[\tau/X]M_1 \equiv [\rho/Y]M_2$  must live in a unique type. Also, that by assumption on either  $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$  or  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , the substitution convention applies giving  $X, Y \notin FV(\tau) \cup FV(\rho)$ .

(In the proof, we will write just “generalizer” instead of “ $Z_0, \dots, Z_h$ -generalizer”).

Case: Assume that  $\Gamma_1 \vdash M_1 : \sigma_1$  by a variable declaration in  $\Gamma_1$ .

Then,  $M_1 \equiv x$  and  $x : \sigma_1 \in \Gamma_1$ .

From the assumption  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , we obtain  $M_2 \equiv x$ .

Furthermore, because  $\Gamma_2 \vdash M_2 : \sigma_2$ , then  $x : \sigma_2 \in \Gamma_2$ .

Take now  $\Gamma_0$  to be the unique generalizer of  $\Gamma_1, \Gamma_2$  by Lemma 5.4,

and  $\sigma_0$  to be the unique generalizer of  $\sigma_1, \sigma_2$  by Type Generalization (Lemma 5.1).

Observe that, by construction,  $x : \sigma_0 \in \Gamma_0$ , from which  $\Gamma_0 \vdash x : \sigma_0$ .

Since  $x$  is clearly the only generalizer of  $M_1 \equiv x$  and  $M_2 \equiv x$ , take  $M_0 \equiv x$ .

Case: Assume that  $\Gamma_1 \vdash M_1 : \sigma_1$  is derived by  $\rightarrow$ -intro.

Then,  $M_1 \equiv \lambda x : \mu_1. M'_1$  and  $\sigma_1 \equiv \mu_1 \rightarrow \rho_1$  with  $\Gamma_1, x : \mu_1 \vdash M'_1 : \rho_1$ .

From  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , we obtain  $M_2 \equiv \lambda x : \mu_2. M'_2$

with  $[\tau/X]\mu_1 = [\rho/Y]\mu_2$  and  $[\tau/X]M'_1 \equiv [\rho/Y]M'_2$ .

Furthermore, because  $\Gamma_2 \vdash M_2 : \sigma_2$ , then  $\sigma_2 \equiv \mu_2 \rightarrow \rho_2$  and  $\Gamma_2, x : \mu_2 \vdash M'_2 : \rho_2$ .

Consider now  $\Gamma_1, x : \mu_1 \vdash M'_1 : \rho_1$  and  $\Gamma_2, x : \mu_2 \vdash M'_2 : \rho_2$ .

By induction, there exist unique generalizers:  $\Gamma'_0$  of  $(\Gamma_1, x : \mu_1), (\Gamma_2, x : \mu_2)$ ;  $M'_0$  of  $M'_1, M'_2$ ; and  $\rho_0$  of  $\rho_1, \rho_2$ , such that  $\Gamma'_0 \vdash M'_0 : \rho_0$ .

But, since generalizers of types and sets of declarations are unique, then  $\Gamma'_0$  must be  $\Gamma_0, x : \mu_0$  where  $\Gamma_0$  and  $\mu_0$  are unique generalizers of  $\Gamma_1, \Gamma_2$  and  $\mu_1, \mu_2$ , respectively.

So, in fact,  $\Gamma_0, x : \mu_0 \vdash M'_0 : \rho_0$ , from which, by  $\rightarrow$ -intro,  $\Gamma_0 \vdash \lambda x : \mu_0. M'_0 : \mu_0 \rightarrow \rho_0$ .

Clearly,  $\lambda x : \mu_0. M'_0$  and  $\mu_0 \rightarrow \rho_0$  are generalizers of  $M_1, M_2$  and  $\sigma_1, \sigma_2$ .

Moreover,  $\mu_0 \rightarrow \rho_0$  is unique by the uniqueness of type generalizers, and  $\lambda x : \mu_0. M'_0$  is unique because any other generalizer of  $M_1, M_2$  would be of the form  $\lambda x : \mu'_0. M''_0$  giving further generalizers,  $\mu'_0$  and  $M''_0$ , of  $\mu_1, \mu_2$  and  $M'_1, M'_2$ , which is impossible.

Hence, take  $M_0 \equiv \lambda x : \mu_0. M'_0$  and  $\sigma_0 \equiv \mu_0 \rightarrow \rho_0$ .

Case: Assume that  $\Gamma_1 \vdash M_1 : \sigma_1$  is derived by  $\rightarrow$ -elim.

Then,  $M_1 \equiv M'_1 N'_1$  with  $\Gamma_1 \vdash M'_1 : \rho_1 \rightarrow \sigma_1$  and  $\Gamma_1 \vdash N'_1 : \rho_1$ .

From  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , we obtain  $M_2 \equiv M'_2 N'_2$   
with  $[\tau/X]M'_1 \equiv [\rho/Y]M'_2$  and  $[\tau/X]N'_1 \equiv [\rho/Y]N'_2$ .

Furthermore, because  $\Gamma_2 \vdash M_2 : \sigma_2$ , then  $\Gamma_2 \vdash M'_2 : \rho_2 \rightarrow \sigma_2$  and  $\Gamma_2 \vdash N'_2 : \rho_2$ .

Consider now  $\Gamma_1 \vdash N'_1 : \rho_1$  and  $\Gamma_2 \vdash N'_2 : \rho_2$ .

By induction, there exist unique generalizers:  $\Gamma_0$  of  $\Gamma_1, \Gamma_2$ ;  $N'_0$  of  $N'_1, N'_2$ ; and  $\rho_0$  of  $\rho_1, \rho_2$ , such that  $\Gamma_0 \vdash N'_0 : \rho_0$ .

Consider also  $\Gamma_1 \vdash M'_1 : \rho_1 \rightarrow \sigma_1$  and  $\Gamma_2 \vdash M'_2 : \rho_2 \rightarrow \sigma_2$ .

By induction, there exist unique generalizers:  $M'_0$  of  $M'_1, M'_2$  and  $\rho'$  of  $\rho_1 \rightarrow \sigma_1, \rho_2 \rightarrow \sigma_2$ , such that  $\Gamma_0 \vdash M'_0 : \rho'$ .

But by the uniqueness of type generalizers,  $\rho'$  must be  $\rho_0 \rightarrow \sigma_0$ , where  $\rho_0$  and  $\sigma_0$  are unique generalizers of  $\rho_1, \rho_2$  and  $\sigma_1, \sigma_2$ , respectively.

Thus, we have  $\Gamma_0 \vdash M'_0 : \rho_0 \rightarrow \sigma_0$  and  $\Gamma_0 \vdash N'_0 : \rho_0$ . So, by  $\rightarrow$ -elim,  $\Gamma_0 \vdash M'_0 N'_0 : \sigma_0$ .

Since  $M'_0 N'_0$  is clearly a generalizer of  $M_1, M_2$ , with uniqueness proven as in the previous case, take  $M_0 \equiv M'_0 N'_0$ .

Case: Assume that  $\Gamma_1 \vdash M_1 : \sigma_1$  is derived by  $\forall$ -intro.

Then,  $M_1 \equiv \lambda Z. M'_1$  and  $\sigma_1 \equiv \forall Z. \mu_1$  with  $\Gamma_1 \vdash M'_1 : \mu_1$ , and  $Z$  not free in the type of any free term variable in  $M'_1$  (by the side-condition on  $\forall$ -intro).

From  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , we obtain  $M_2 \equiv \lambda Z. M'_2$  with  $[\tau/X]M'_1 \equiv [\rho/Y]M'_2$ .

Furthermore, because  $\Gamma_2 \vdash M_2 : \sigma_2$ , then  $\sigma_2 \equiv \forall Z. \mu_2$  and  $\Gamma_2 \vdash M'_2 : \mu_2$   
with  $Z$  not free in the type of any free term variable in  $M'_2$ .

Consider now  $\Gamma_1 \vdash M'_1 : \mu_1$  and  $\Gamma_2 \vdash M'_2 : \mu_2$ .

By induction, there exist unique generalizers:  $\Gamma_0$  of  $\Gamma_1, \Gamma_2$ ;  $M'_0$  of  $M'_1, M'_2$ ; and  $\mu_0$  of  $\mu_1, \mu_2$ , such that  $\Gamma_0 \vdash M'_0 : \mu_0$ .

Observe now that  $Z$  is not free in the type of any free term variable in  $M'_0$ , since, by the definition of generalizer,  $M'_0$  contains exactly the free term variables of  $M'_1, M'_2$ .

Thus, we can apply  $\forall$ -intro to  $\Gamma_0 \vdash M'_0 : \mu_0$  to obtain  $\Gamma_0 \vdash \lambda Z. M'_0 : \forall Z. \mu_0$ .

Clearly,  $\lambda Z. M'_0$  and  $\forall Z. \mu_0$  are generalizers of  $M_1, M_2$  and  $\sigma_1, \sigma_2$ , respectively. Their uniqueness follows as before. Hence, take  $M_0 \equiv \lambda Z. M'_0$  and  $\sigma_0 \equiv \forall Z. \mu_0$ .

Case: Assume that  $\Gamma_1 \vdash M_1 : \sigma_1$  is derived by  $\forall$ -elim.

Then,  $M_1 \equiv M'_1 \mu_1$  and  $\sigma_1 \equiv [\mu_1/Z] \rho_1$  with  $\Gamma_1 \vdash M'_1 : \forall Z. \rho_1$ .

From  $[\tau/X]M_1 \equiv [\rho/Y]M_2$ , we obtain  $M_2 \equiv M'_2 \mu_2$

with  $[\tau/X]M'_1 \equiv [\rho/Y]M'_2$  and  $[\tau/X]\mu_1 \equiv [\rho/Y]\mu_2$ .

Furthermore, since  $\Gamma_2 \vdash M_2 : \sigma_2$ , then  $\Gamma_2 \vdash M'_2 : \forall Z. \rho_2$  and  $\sigma_2 \equiv [\mu_2/Z] \rho_2$ .

Consider now  $\Gamma_1 \vdash M'_1 : \forall Z. \rho_1$  and  $\Gamma_2 \vdash M'_2 : \forall Z. \rho_2$ .

By induction, there exist unique generalizers:  $\Gamma_0$  of  $\Gamma_1, \Gamma_2$ ;  $M'_0$  of  $M'_1, M'_2$ ; and  $\rho'$  of  $\forall Z. \rho_1, \forall Z. \rho_2$ , such that  $\Gamma_0 \vdash M'_0 : \rho'$ .

By unicity of type generalizers,  $\rho' \equiv \forall Z. \rho_0$ , where  $\rho_0$  is the generalizer of  $\rho_1, \rho_2$ .

Thus, we have  $\Gamma_0 \vdash M'_0 : \forall Z. \rho_0$ , from which, by  $\forall$ -elim,  $\Gamma_0 \vdash M'_0 \mu_0 : [\mu_0/Z] \rho_0$   
where  $\mu_0$  is the unique generalizer of  $\mu_1, \mu_2$  by Type Generalization (Lemma 5.1).

Clearly,  $M'_0 \mu_0$  is a generalizer of  $M_1, M_2$ , with uniqueness proven as before.

Furthermore, by Lemma 5.3,  $[\mu_0/Z] \rho_0$  is the unique generalizer of  $\sigma_1 \equiv [\mu_1/Z] \rho_1$ ,

$\sigma_2 \equiv [\mu_2/Z] \rho_2$ . Hence, take  $M_0 \equiv M'_0 \mu_0$  and  $\sigma_0 \equiv [\mu_0/Z] \rho_0$ . ■

## 6 Weak Genericity of F-equality

In this section, we prove a weak form of Genericity that will be used in the final proof. The weakness or asymmetry arises because  $=_F$  is used in the premise instead of  $=_{Fc}$ . Generalizers are a key tool in the proof. We first need the following lemma about simultaneous substitutions.

### Lemma 6.1

Given type  $\sigma$ , if  $[\tau_1/X_1, \dots, \tau_n/X_n] \sigma = [\rho_1/X_1, \dots, \rho_n/X_n] \sigma$  and  $\tau_i \neq \rho_i$  for some  $1 \leq i \leq n$ , then  $X_i$  is not free in  $\sigma$ .

Proof: by induction on the structure of  $\sigma$ . Note that the substitution convention is used to assume that  $X_1, \dots, X_n$  are not free in  $\tau_1, \dots, \tau_n, \rho_1, \dots, \rho_n$ . ■

### Theorem 6.2 (Weak Genericity of F-equality)

Let  $\Gamma \vdash M_1, M_2 : \sigma$ . If  $[\tau/X]M_1 =_F [\tau/X]M_2$  for some type  $\tau$ , then  $M_1 =_{Fc} M_2$ .

Proof: Let  $M'_1$  and  $M'_2$  be the normal forms of  $M_1$  and  $M_2$ .

Then,  $\Gamma \vdash M'_1, M'_2 : \sigma$  since normalization preserves typing.

Further, since reduction is type-substitutive<sup>1</sup>, and since type substitution preserves normal forms, then, from  $[\tau/X]M_1 =_F [\tau/X]M_2$ , we obtain  $[\tau/X]M'_1 \equiv [\tau/X]M'_2$ .

We now apply Term Generalization to

$$[\tau/X]M'_1 \equiv [\tau/X]M'_2 \quad (3)$$

We are in the situation  $\tau = \rho$  so the first case of the definition of generalizer applies, i.e.  $h = 1$ . Thus, choose fresh type variables  $Z_0, Z_1, Z_2$ .

By Term Generalization (Theorem 5.5), there exist unique  $Z_0, Z_1, Z_2$ -generalizers:  $\Gamma_0$  of  $\Gamma, \Gamma$ ;  $M'_0$  of  $M'_1, M'_2$ ; and  $\sigma_0$  of  $\sigma, \sigma$ , such that  $\Gamma_0 \vdash M'_0 : \sigma_0$ .

By the definition of generalizer, we have

$$\begin{aligned} [X/Z_0, \tau/Z_1, X/Z_2] \Gamma_0 &= \Gamma = [\tau/Z_0, X/Z_1, X/Z_2] \Gamma_0 \\ [X/Z_0, \tau/Z_1, X/Z_2] \sigma_0 &= \sigma = [\tau/Z_0, X/Z_1, X/Z_2] \sigma_0 \end{aligned}$$

Now, by the substitution convention applied to (3),  $X \notin FV(\tau)$ .

So, certainly,  $\tau \neq X$ . We can thus apply Lemma 6.1 to the above two equations to obtain  $Z_0$  and  $Z_1$  not free in  $\Gamma_0$  and  $\sigma_0$ .

Hence, we can apply Axiom C\* to  $M'_0$  for  $Z_0, Z_1$  in the following:

$$\begin{aligned} M_1 &=_F M'_1 && M'_1 \text{ is the normal form of } M_1 \\ &\equiv [X/Z_0, \tau/Z_1, X/Z_2] M'_0 && M'_0 \text{ is the generalizer of } M'_1, M'_2 \\ &=_F [\tau/Z_0, X/Z_1, X/Z_2] M'_0 && \text{by Axiom C*} \\ &\equiv M'_2 && M'_0 \text{ is the generalizer of } M'_1, M'_2 \\ &=_F M_2 && M'_2 \text{ is the normal form of } M_2 \end{aligned} \quad \blacksquare$$

<sup>1</sup>If  $M$  reduces to  $M'$  then  $[\tau/X]M$  reduces to  $[\tau/X]M'$  (cf. [Bar84, page 55]).

## 7 Quasi-Genericity of $C^*$ -equality

This section shows that applications of Axiom  $C^*$  preserve the type substitution structure of terms. That is, if Axiom  $C^*$  is applied to a term of the form  $[\tau/X]M$ , then the result is a term of the form  $[\tau/X]N$  with  $M =_{C^*} N$ . We call this property *Quasi-Genericity* of  $C^*$ -equality (since it resembles genericity), and the proof of this uses generalizers.

We will write  $M \stackrel{1}{=}_{C^*} N$  if  $M$  and  $N$  are made equal by one application of Axiom  $C^*$  only, and  $M =_{C^*} N$  if Axiom  $C^*$  is applied zero or more times. Clearly, if  $M \stackrel{1}{=}_{C^*} N$ , then the single application of Axiom  $C^*$  may have been made either to a proper subterm of  $M$ , or to the entire term  $M$ . Note, however, that an application of Axiom  $C^*$  to a term cannot always be split into applications to subterms, as the example of Section 4 shows.

### Theorem 7.1 (Quasi-Genericity of $C^*$ -equality)

If  $[\tau/X]M =_{C^*} N'$  then there exists a term  $N$  such that  $M =_{C^*} N$  and  $[\tau/X]N \equiv N'$ .

*Proof:* Construct  $N$  by induction on the number of  $C^*$ -applications in  $[\tau/X]M =_{C^*} N'$ .

Clearly, if there are 0 applications, i.e.,  $[\tau/X]M \equiv N'$ , then take  $N \equiv M$ .

We consider here only the case,  $[\tau/X]M \stackrel{1}{=}_{C^*} N'$ , as the inductive case is obvious by transitivity.

Assume thus that  $[\tau/X]M \stackrel{1}{=}_{C^*} N'$ . Then, as remarked above, Axiom  $C^*$  is applied either to a proper subterm of  $[\tau/X]M$ , or to  $[\tau/X]M$  itself.

If Axiom  $C^*$  is applied to a proper subterm of  $[\tau/X]M$ , the theorem is proven by straightforward induction on the structure of  $M$ .

Consider then the case when Axiom  $C^*$  is applied to  $[\tau/X]M$  itself.

We assume, with no loss of generality, that, by variable renaming,  $X \notin FV(N')$ .

Then, by the definition of Axiom  $C^*$ , there exists a term  $M'$ , types  $\rho, \rho'$ , and a type variable  $Y$ , such that

$$[\tau/X]M \equiv [\rho/Y]M' \stackrel{1}{=}_{C^*} [\rho'/Y]M' \equiv N' \quad (4)$$

where, for  $\Gamma \vdash M : \sigma$ , we have  $\Gamma \vdash M' : \sigma'$ , and  $Y$  not free in  $\Gamma$  nor  $\sigma'$ .

Since Axiom  $C^*$  is actually applied, then  $Y \in FV(M')$  and, thus,  $X \notin FV(\rho')$ , else  $X$  would be free in  $N'$ , against the assumption.

We now apply Term Generalization to  $[\tau/X]M \equiv [\rho/Y]M'$ .

Case: Assume that  $\tau \text{ in}_k \rho$  for  $k > 0$ .

Choose fresh type variables  $Z_0, \dots, Z_h$  where  $h = 2^k$ .

By Term Generalization (Theorem 5.5), there exist unique  $Z_0, \dots, Z_h$ -generalizers:  $\Gamma_0$  of  $\Gamma, \Gamma$ ;  $M_0$  of  $M, M'$ ; and  $\sigma_0$  of  $\sigma, \sigma'$ , such that  $\Gamma_0 \vdash M_0 : \sigma_0$ .

Observe now that, by the definition of generalizer, we have

$$\Gamma = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \Gamma_0 \text{ and } \sigma' = [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] \sigma_0$$

But since we also have  $Y$  not free in  $\Gamma$  or  $\sigma'$ , then  $Z_1, \dots, Z_h$  cannot be free in  $\Gamma_0$  or  $\sigma_0$ .

Hence, since  $\Gamma_0 \vdash M_0 : \sigma_0$ , we can apply Axiom  $C^*$  to  $M_0$  for the variables  $Z_1, \dots, Z_h$ .

Thus, if we take

$$N \equiv [X/Z_0, \rho'/Z_1, \dots, \rho'/Z_h] M_0$$

we get the desired result, as

$$\begin{aligned} M &\equiv [X/Z_0, \rho_1^X/Z_1, \dots, \rho_h^X/Z_h] M_0 && M_0 \text{ is the generalizer of } M, M' \\ &=_{c^*} [X/Z_0, \rho'/Z_1, \dots, \rho'/Z_h] M_0 && \text{by Axiom } C^* \\ &\equiv N \end{aligned}$$

and

$$\begin{aligned} [\tau/X] N &\equiv [\tau/Z_0, \rho'/Z_1, \dots, \rho'/Z_h] M_0 && \text{since } X \notin FV(\rho') \\ &\equiv [\rho'/Y] [\tau/Z_0, Y/Z_1, \dots, Y/Z_h] M_0 && \text{by rearranging substitutions} \\ &\equiv [\rho'/Y] M' && M_0 \text{ is the generalizer of } M, M' \\ &\equiv N' && \text{by (4)} \end{aligned}$$

Case: Assume that  $\rho \text{ in}_k \tau$  for  $k \geq 0$  and the previous case does not apply.

Choose fresh type variables  $Z_0, \dots, Z_h$  where  $h = 2^k$ .

By Term Generalization (Theorem 5.5), there exist unique  $Z_0, \dots, Z_h$ -generalizers:  $\Gamma_0$  of  $\Gamma, \Gamma$ ;  $M_0$  of  $M, M'$ ; and  $\sigma_0$  of  $\sigma, \sigma'$ , such that  $\Gamma_0 \vdash M_0 : \sigma_0$ .

Observe now that, by the definition of generalizer,

we have  $\Gamma = [Y/Z_0, \tau/Z_1, \tau_2^Y/Z_2, \dots, \tau_h^Y/Z_h] \Gamma_0$

and  $\sigma' = [Y/Z_0, \tau/Z_1, \tau_2^Y/Z_2, \dots, \tau_h^Y/Z_h] \sigma_0$ .

But, we also have that  $Y$  is not free in  $\Gamma$  or  $\sigma'$ ,

so  $Z_0, Z_2, \dots, Z_h$  cannot be free in  $\Gamma_0$  or  $\sigma_0$ .

Hence, since  $\Gamma_0 \vdash M_0 : \sigma_0$ , we can apply Axiom  $C^*$  to  $M_0$  for  $Z_0, Z_2, \dots, Z_h$ .

Let  $\tau'_i \equiv [\rho'/Y] \tau_i^Y$ . Then, if we take

$$N \equiv [\rho'/Z_0, X/Z_1, \tau'_2/Z_2, \dots, \tau'_h/Z_h] M_0$$

we get the desired result, as

$$\begin{aligned} M &\equiv [\rho/Z_0, X/Z_1, X/Z_2, \dots, X/Z_h] M_0 && M_0 \text{ is the generalizer of } M, M' \\ &=_{c^*} [\rho'/Z_0, X/Z_1, \tau'_2/Z_2, \dots, \tau'_h/Z_h] M_0 && \text{by Axiom } C^* \\ &\equiv N \end{aligned}$$

and

$$\begin{aligned} [\tau/X] N &\equiv [\rho'/Z_0, \tau/Z_1, \tau'_2/Z_2, \dots, \tau'_h/Z_h] M_0 && \text{since } X \notin FV(\rho') \\ &\equiv [\rho'/Y] [Y/Z_0, \tau/Z_1, \tau_2^Y/Z_2, \dots, \tau_h^Y/Z_h] M_0 && \\ & && \text{by rearranging substitutions} \\ &\equiv [\rho'/Y] M' && M_0 \text{ is the generalizer of } M, M' \\ &\equiv N' && \text{by (4)} \end{aligned} \quad \blacksquare$$

The next theorem is another weak form of Genericity, with  $C^*$ -equality in the premise instead of  $Fc$ -equality. Quasi-Genericity is used in the proof.



**Theorem 7.2 (Weak-Genericity of C\*-equality)**

Let  $\Gamma \vdash M_1, M_2 : \sigma$ . If  $[\tau/X]M_1 =_{c^*} [\tau/X]M_2$  for some type  $\tau$ , then  $M_1 =_{Fc} M_2$ .

Proof: Apply Quasi-Genericity of C\*-equality (Theorem 7.1) to  $[\tau/X]M_1 =_{c^*} [\tau/X]M_2$ .

Thus, there exists a term  $N$  such that  $M_1 =_{Fc} N$  and  $[\tau/X]N \equiv [\tau/X]M_2$ .

Observe that, since  $M_1 =_{Fc} N$ , then  $N$  must live in  $\sigma$ , the type of  $M_1$  and  $M_2$ .

Apply now Weak Genericity of F-equality (Theorem 6.2) to  $[\tau/X]N \equiv [\tau/X]M_2$ .

Then,  $N =_{Fc} M_2$ . Hence,  $M_1 =_{Fc} N =_{Fc} M_2$ . ■

Note that the property of preserving type substitution structure does **not** hold for F-equality. Backward  $\beta_2$  reduction causes problems as witnessed by the following counter-example. Assume  $x$  has type  $\forall Y.Y$ . Take  $M \equiv xX$  with  $\tau \equiv \sigma_1 \rightarrow \sigma_2$  and  $N' \equiv (\lambda Z.x(Z \rightarrow \sigma_2))\sigma_1$ . Then,

$$[\tau/X]M \equiv x(\sigma_1 \rightarrow \sigma_2) \quad \beta_2 \longleftarrow \quad (\lambda Z.x(Z \rightarrow \sigma_2))\sigma_1 \equiv N'$$

Now, since  $\tau \equiv \sigma_1 \rightarrow \sigma_2$  does not occur in  $N'$ , then any  $N$  such that  $[\tau/X]N \equiv N'$  cannot contain  $X$  free. Thus,  $N \equiv [\tau/X]N \equiv N'$ , and  $N$  has type  $\tau$ . But  $M$  has type  $X$ . Hence,  $M = N$  is impossible since they live in different types.

However, all forward reductions preserve type substitution structure, as does backward  $\eta_2$  reduction. Proofs of these are straightforward.

**Fact 7.3**

If  $[\tau/X]M \longrightarrow_F N'$  then there exists a term  $N$  such that  $M \longrightarrow_F N$  and  $[\tau/X]N \equiv N'$ .

**Fact 7.4**

If  $[\tau/X]M \eta_2 \longleftarrow N'$  then there exists a term  $N$  such that  $M \eta_2 \longleftarrow N$  and  $[\tau/X]N \equiv N'$ .

**8 Commutativity of C\*-equality with Reduction**

This section describes the commutativity of C\*-equality with reduction. It turns out that C\*-equality commutes with  $\beta_1$ ,  $\beta_2$ , and  $\eta_1$  reductions but **not** with  $\eta_2$  reduction. To see this last point, take  $M$  of type  $\forall Z.\sigma$  with  $Z \notin FV(\sigma)$ , and  $X$  fresh. Then, because  $\lambda X.M\tau$  does not  $\eta_2$ -reduce to  $M$ , we cannot complete the following diagram:

$$\begin{array}{ccc} \lambda X.MX & =_c & \lambda X.M\tau \\ \downarrow \eta_2 & & \\ M & & \end{array}$$

We need the following lemma about the substitutivity of  $C^*$ -equality.

**Lemma 8.1 (Substitutivity of  $C^*$ -equality)**

If  $M_1 =_{C^*} M_2$  and  $N_1 =_{C^*} N_2$  then  $[N_1/x]M_1 =_{C^*} [N_2/x]M_2$  and  $[\tau/X]M_1 =_{C^*} [\tau/X]M_2$ .

Proof: An easy induction on the structure of  $M_1$ . ■

We now prove that  $C^*$ -equality commutes with  $\beta_1\beta_2\eta_1$  reduction, first for the one-step case, then for the multi-step case. Note that, in the one-step case, a multi-step  $C^*$ -equality completes the commuting diagram.

**Lemma 8.2 (One-Step Commutativity)**

If  $M \stackrel{1}{=}_{C^*} N$  then there exists a term  $N'$  such that

$$\begin{array}{ccc} M & \stackrel{1}{=}_{C^*} & N \\ \beta_1\beta_2\eta_1 \downarrow 1 & & \downarrow 1 \\ M' & =_{C^*} & N' \end{array}$$

Proof: By case analysis of  $M \xrightarrow{1}_{\beta_1\beta_2\eta_1} M'$  and  $M \stackrel{1}{=}_{C^*} N$ .

Since  $\beta_1\beta_2\eta_1$  is substitutive, we can assume that the reduction is applied directly to  $M$ , ignoring the cases where it is applied to a subterm or superterm of  $M$ .

Case:  $(\lambda x : \mu. M_1)M_2 \xrightarrow{1}_{\beta_1} [M_2/x]M_1$ .

*Subcase:* Assume that Axiom  $C^*$  is applied to  $M_1$ .

Then,  $M_1 \stackrel{1}{=}_{C^*} N_1$  and  $(\lambda x : \mu. M_1)M_2 \stackrel{1}{=}_{C^*} (\lambda x : \mu. N_1)M_2$ .

Clearly,  $(\lambda x : \mu. N_1)M_2 \xrightarrow{1}_{\beta_1} [M_2/x]N_1$ .

And, by Lemma 8.1,  $[M_2/x]M_1 =_{C^*} [M_2/x]N_1$ .

Therefore, take  $N' \equiv [M_2/x]N_1$ .

*Subcase:* Assume that Axiom  $C^*$  is applied to  $M_2$ .

Then,  $M_2 \stackrel{1}{=}_{C^*} N_2$  and  $(\lambda x : \mu. M_1)M_2 \stackrel{1}{=}_{C^*} (\lambda x : \mu. M_1)N_2$ .

Clearly,  $(\lambda x : \mu. M_1)N_2 \xrightarrow{1}_{\beta_1} [N_2/x]M_1$ .

And, by Lemma 8.1,  $[M_2/x]M_1 =_{C^*} [N_2/x]M_1$ .

Therefore, take  $N' \equiv [N_2/x]M_1$ .

*Subcase:* Assume that Axiom  $C^*$  is applied to  $\lambda x : \mu. M_1$ .

Then, by the definition of Axiom  $C^*$ , there exist  $\nu, N_1, \rho, \rho', Y$  such that

$\lambda x : \mu. M_1 \equiv [\rho/Y](\lambda x : \nu. N_1) \stackrel{1}{=}_{C^*} [\rho'/Y](\lambda x : \nu. N_1)$

with  $\mu = [\rho/Y]\nu$  and  $M_1 \equiv [\rho/Y]N_1$ ,

and  $\Gamma \vdash \lambda x : \nu. N_1 : \nu \rightarrow \sigma$ , and  $Y$  not free in  $\Gamma$  or  $\nu \rightarrow \sigma$ .

Clearly,  $Y$  is also not free in  $\nu$ . Hence,  $\mu = [\rho/Y]\nu = \nu$ .

Moreover,  $Y$  is not free in  $\sigma$ , the type of  $N_1$ .

Therefore, Axiom  $C^*$  is applied to  $M_1 \equiv [\rho/Y]N_1$ , and that subcase applies.

*Subcase:* Assume that Axiom C\* is applied to  $(\lambda x : \mu.M_1)M_2$ .

Then, by the definition of Axiom C\*, there exist  $\nu, N_1, N_2, \rho, \rho', Y$  such that

$$(\lambda x : \mu.M_1)M_2 \equiv [\rho/Y]((\lambda x : \nu.N_1)N_2) \stackrel{1}{=}_{c^*} [\rho'/Y]((\lambda x : \nu.N_1)N_2)$$

with  $\mu = [\rho/Y]\nu$ ,  $M_1 \equiv [\rho/Y]N_1$ ,  $M_2 \equiv [\rho/Y]N_2$ ,

and  $\Gamma \vdash (\lambda x : \nu.N_1)N_2 : \sigma$ , and  $Y$  not free in  $\Gamma$  or  $\sigma$ .

Since  $\Gamma \vdash (\lambda x : \nu.N_1)N_2 : \sigma$ , then  $\Gamma \vdash [N_2/x]N_1 : \sigma$ .

Axiom C\* can thus be applied to  $[N_2/x]N_1$ .

Hence, take  $N' \equiv [\rho'/Y][N_2/x]N_1$ , for then

$$\begin{aligned} [M_2/x]M_1 &\equiv [\rho/Y][N_2/x]N_1 \quad \text{since } M_1 \equiv [\rho/Y]N_1 \text{ and } M_2 \equiv [\rho/Y]N_2 \\ &\equiv_{c^*} [\rho'/Y][N_2/x]N_1 \quad \text{by Axiom C}^* \end{aligned}$$

and  $[\rho'/Y]((\lambda x : \nu.N_1)N_2) \xrightarrow{1}_{\beta_1} [\rho'/Y][N_2/x]N_1$ , since  $\beta_1$  is substitutive.

Case:  $(\lambda X.M_1)\mu \xrightarrow{1}_{\beta_2} [\mu/X]M_1$ .

*Subcase:* Assume that Axiom C\* is applied to  $M_1$ .

Then,  $M_1 \stackrel{1}{=}_{c^*} N_1$  and  $(\lambda X.M_1)\mu \stackrel{1}{=}_{c^*} (\lambda X.N_1)\mu$ .

Clearly,  $(\lambda X.N_1)\mu \xrightarrow{1}_{\beta_2} [\mu/X]N_1$ .

And, by Lemma 8.1,  $[\mu/X]M_1 \equiv_{c^*} [\mu/X]N_1$ .

Therefore, take  $N' \equiv [\mu/X]N_1$ .

*Subcase:* Assume that Axiom C\* is applied to  $\lambda X.M_1$ .

Then, by the definition of Axiom C\*, there exist  $N_1, \rho, \rho', Y$  such that

$$\lambda X.M_1 \equiv [\rho/Y](\lambda X.N_1) \stackrel{1}{=}_{c^*} [\rho'/Y](\lambda X.N_1)$$

with  $M_1 \equiv [\rho/Y]N_1$ , and  $\Gamma \vdash \lambda X.N_1 : \forall X.\sigma$ , and  $Y$  not free in  $\Gamma$  or  $\forall X.\sigma$ .

Clearly,  $Y$  is not free in  $\sigma$ , the type of  $N_1$ .

Axiom C\* is therefore applied to  $M_1 \equiv [\rho/Y]N_1$  and that subcase applies.

*Subcase:* Assume that Axiom C\* is applied to  $(\lambda X.M_1)\mu$ .

Then, by the definition of Axiom C\*, there exist  $N_1, \nu, \rho, \rho', Y$  such that

$$(\lambda X.M_1)\mu \equiv [\rho/Y]((\lambda X.N_1)\nu) \stackrel{1}{=}_{c^*} [\rho'/Y]((\lambda X.N_1)\nu)$$

with  $M_1 \equiv [\rho/Y]N_1$  and  $\mu = [\rho/Y]\nu$ ,

and  $\Gamma \vdash (\lambda X.N_1)\nu : \sigma$ , and  $Y$  not free in  $\Gamma$  or  $\sigma$ .

Since  $\Gamma \vdash (\lambda X.N_1)\nu : \sigma$ , then  $\Gamma \vdash [\nu/X]N_1 : \sigma$ .

Axiom C\* can thus be applied to  $[\nu/X]N_1$ .

Hence, take  $N' \equiv [\rho'/Y][\nu/X]N_1$ , for then

$$\begin{aligned} [\mu/X]M_1 &\equiv [\rho/Y][\nu/X]N_1 \quad \text{since } \mu \equiv [\rho/Y]\nu \text{ and } M_1 \equiv [\rho/Y]N_1 \\ &\equiv_{c^*} [\rho'/Y][\nu/X]N_1 \quad \text{by Axiom C}^* \end{aligned}$$

and  $[\rho'/Y]((\lambda X.N_1)\nu) \xrightarrow{1}_{\beta_2} [\rho'/Y][\nu/X]N_1$ , since  $\beta_2$  is substitutive.

Case:  $\lambda x : \mu.M_1 x \xrightarrow{1}_{\eta_1} M_1$  with  $x$  not free in  $M_1$ .

*Subcase:* Assume that Axiom C\* is applied to  $M_1$ .

Then,  $M_1 \stackrel{1}{=}_{c^*} N_1$  and  $\lambda x : \mu.M_1 x \stackrel{1}{=}_{c^*} \lambda x : \mu.N_1 x$ .

Now, since  $x$  is not free in  $M_1$  and since Axiom C\* does not affect term variables,

then  $x$  is also not free in  $N_1$ . Thus,  $\lambda x : \mu.N_1 x \xrightarrow{1}_{\eta_1} N_1$ .

Therefore, take  $N' \equiv N_1$ .

*Subcase:* Assume that Axiom  $C^*$  is applied to  $M_1x$ .

Then, by the definition of Axiom  $C^*$ , there exist  $N_1, \rho, \rho', Y$  such that

$$M_1x \equiv [\rho/Y](N_1x) \stackrel{1}{=}_{C^*} [\rho'/Y](N_1x)$$

with  $M_1 = [\rho/Y]N_1$ , and  $\Gamma, x:\mu \vdash N_1x : \sigma$ , and  $Y$  not free in  $\Gamma, x:\mu$  or  $\sigma$ .

Clearly,  $Y$  is also not free in  $\mu \rightarrow \sigma$ , the type of  $N_1$ .

Axiom  $C^*$  is therefore applied to  $M_1 \equiv [\rho/Y]N_1$  and that subcase applies.

*Subcase:* Assume that Axiom  $C^*$  is applied to  $\lambda x:\mu.M_1x$ .

Then, by the definition of Axiom  $C^*$ , there exist  $\nu, N_1, \rho, \rho', Y$  such that

$$\lambda x:\mu.M_1x \equiv [\rho/Y](\lambda x:\nu.N_1x) \stackrel{1}{=}_{C^*} [\rho'/Y](\lambda x:\nu.N_1x)$$

with  $\mu = [\rho/Y]\nu$  and  $M_1 = [\rho/Y]N_1$ ,

and  $\Gamma \vdash \lambda x:\nu.N_1x : \nu \rightarrow \sigma$ , and  $Y$  not free in  $\Gamma$  or  $\nu \rightarrow \sigma$ .

$Y$  is therefore not free in  $\nu$ , so,  $\mu = [\rho/Y]\nu = \nu$ .

Also,  $Y$  is not free in  $\sigma$ , the type of  $N_1x$ .

Axiom  $C^*$  is thus applied to  $M_1x \equiv [\rho/Y](N_1x)$ , and that subcase applies. ■

### Theorem 8.3 (Commutativity)

$$\begin{array}{ccc} \text{If} & M \stackrel{1}{=}_{C^*} N & \text{then there exists a term } N' \text{ such that} \\ & \downarrow \beta_1\beta_2\eta_1 & \downarrow \beta_1\beta_2\eta_1 \\ & M' & M' \stackrel{1}{=}_{C^*} N' \end{array}$$

**Proof:** By decomposing the multi-step  $C^*$ -equalities and  $\beta_1\beta_2\eta_1$ -reductions into single steps, and using One-Step Commutativity (Lemma 8.2) to complete the following diagram:

$$\begin{array}{ccccccc} M & \stackrel{1}{=}_{C^*} & N_1 & \stackrel{1}{=}_{C^*} & \dots & \stackrel{1}{=}_{C^*} & N \\ \downarrow \beta_1\beta_2\eta_1 & & \downarrow \beta_1\beta_2\eta_1 & & & & \downarrow \beta_1\beta_2\eta_1 \\ M'_1 & \stackrel{1}{=}_{C^*} & N_{11} & \stackrel{1}{=}_{C^*} \dots \stackrel{1}{=}_{C^*} & N_{1i} & \dots & \\ \downarrow \beta_1\beta_2\eta_1 & & \downarrow \beta_1\beta_2\eta_1 & & & & \vdots \\ M'_2 & \stackrel{1}{=}_{C^*} \dots \stackrel{1}{=}_{C^*} & N_{2j} & \dots & & & \\ \vdots & & & & & & \vdots \\ \downarrow \beta_1\beta_2\eta_1 & & & & & & \downarrow \beta_1\beta_2\eta_1 \\ M' \stackrel{1}{=}_{C^*} \dots & & \dots & & \dots & & \stackrel{1}{=}_{C^*} N' \end{array}$$

■

## 9 The Genericity Theorem

Finally, in this section, we prove the Main Lemma that leads to the Genericity Theorem. We first need the following lemma:

### Lemma 9.1 ( $\eta_2$ -postponement)

If  $M \rightarrow_F M'$  then there exists a term  $M''$  such that  $M \rightarrow_{\beta_1\beta_2\eta_1} M'' \rightarrow_{\eta_2} M'$ .

Proof: Easy; see [BS93]. ■

### Lemma 9.2 (Main)

Let  $\Gamma \vdash M, N : \sigma$ . If  $[\tau/X]M =_{Fc} [\tau/X]N$  for some type  $\tau$ , then  $M =_{Fc} N$ .

Proof: Observe first that the chain of Fc-equalities from  $[\tau/X]M$  to  $[\tau/X]N$  can be written:

$$[\tau/X]M =_F M''_1 =_{c^*} M''_2 =_F M''_3 =_{c^*} \dots =_F M''_{n-1} =_{c^*} M''_n =_F [\tau/X]N$$

that is, as alternations of F-equalities and  $C^*$ -equalities with the initial and final equalities being F-equalities. These initial or final F-equalities may be just trivial syntactic identities if, in fact, a  $C^*$ -equality starts or ends the chain.

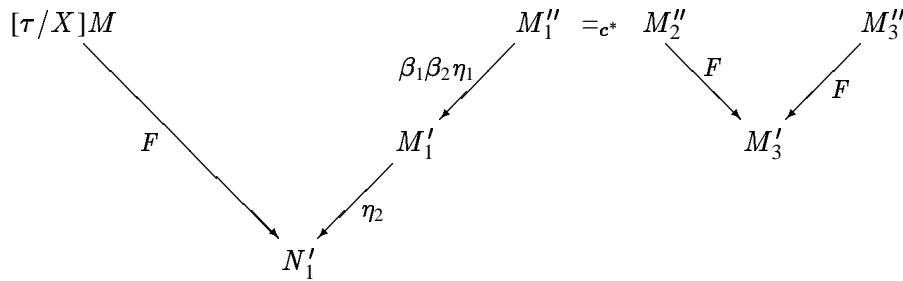
Case: The chain consists entirely of F-equalities, i.e.,  $[\tau/X]M =_F [\tau/X]N$ . Then, by Weak Genericity of F-equality (Theorem 6.2), we have the result  $M =_{Fc} N$ .

Case: The chain consists entirely of  $C^*$ -equalities, i.e.,  $[\tau/X]M =_{c^*} [\tau/X]N$ . Then, by Weak Genericity of  $C^*$ -equality (Theorem 7.2),  $M =_{Fc} N$ .

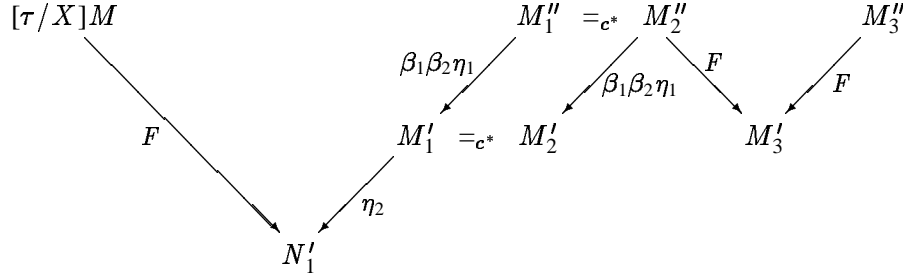
Case: There is at least one (non-trivial)  $C^*$ -equality and one (non-trivial) F-equality. We proceed with a series of transformations on the chain, starting with the first three links:

$$[\tau/X]M =_F M''_1 =_{c^*} M''_2 =_F M''_3$$

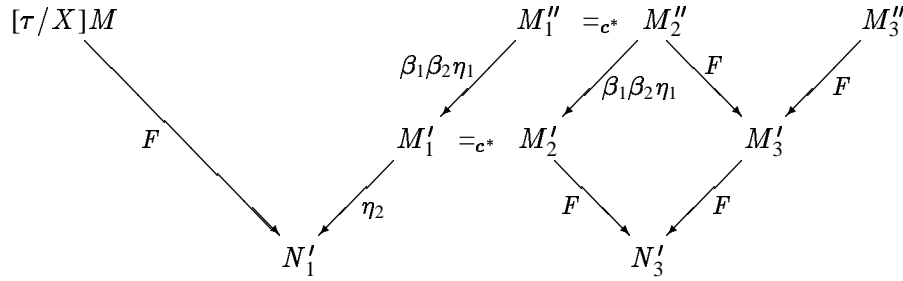
First, as a consequence of the equational Church-Rosser property for F, transform the F-equalities into reductions. Then, apply  $\eta_2$ -postponement (Lemma 9.1) to the reduction sequence from  $M''_1$ . Thus, there exist terms  $M'_1, M'_3, N'_1$  such that:



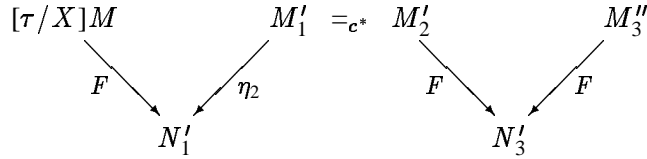
Then, by Commutativity of  $C^*$ -equality with  $\beta_1\beta_2\eta_1$  reduction (Theorem 8.3), there exists  $M'_2$  such that



The Church-Rosser property can then be used to complete the diamond between  $M'_2$  and  $M'_3$ :



In this way, the original three links from  $[\tau/X]M$  to  $M''_3$  can be replaced by;



Repeat this transformation down the rest of the chain by sets of three consecutive links of the form  $\bullet =_F \bullet =_{c^*} \bullet =_F \bullet$  continuing with  $M'_2 =_F M'_3 =_{c^*} M'_4 =_F M'_5$ . Note that the first link of each set coincides with the last link of the previously modified set. At the end, the transformed chain will look like:



where each left-pointing arrow, except for the final one, consists of forward  $\eta_2$  reductions. The final left-pointing arrow, and all the right-pointing ones, consist of forward  $\beta_1\beta_2\eta_1\eta_2$  reductions.

From here on, we work with the transformed chain. Consider now the start of it:

$$\begin{array}{ccc} [\tau/X]M & & M'_1 =_{c^*} M'_2 \\ & \searrow F \quad \swarrow \eta_2 & \\ & N'_1 & \end{array}$$

- By Fact 7.3, there exists  $N_1$  such that  $N'_1 \equiv [\tau/X]N_1$  and  $M \twoheadrightarrow_F N_1$ .
- By Fact 7.4, there exists  $M_1$  such that  $M'_1 \equiv [\tau/X]M_1$  and  $N_1 \eta_2 \multimap M_1$ .
- By Quasi-Genericity of  $C^*$ -equality (Theorem 7.1), there exists  $M_2$  such that  $M'_2 \equiv [\tau/X]M_2$  and  $M_1 =_{c^*} M_2$ .

Thus, we have

$$\begin{array}{ccc} [\tau/X]M & & [\tau/X]M_1 \equiv M'_1 =_{c^*} M'_2 \equiv [\tau/X]M_2 \\ & \searrow F \quad \swarrow \eta_2 & \\ & N'_1 \equiv [\tau/X]N_1 & \end{array}$$

with  $M \twoheadrightarrow_F N_1 \eta_2 \multimap M_1 =_{c^*} M_2$ . Hence,  $M =_{Fc} M_2$ .

Now, iterate this process along the chain from  $M'_2 \equiv [\tau/X]M_2$ .

We thus “push” the type substitution  $[\tau/X]$  along the chain so that, eventually, for  $M'_n$ , the penultimate term of the chain, there exists a term  $M_n$  such that  $M'_n \equiv [\tau/X]M_n$  and  $M =_{Fc} M_n$ . Apply then Weak Genericity of F-equality (Theorem 6.2) to the last link  $[\tau/X]M_n \equiv M'_n =_F [\tau/X]N$ . This gives  $M_n =_{Fc} N$ .

Since  $M =_{Fc} M_n$ , then  $M =_{Fc} N$  as required. ■

### Theorem 9.3 (Genericity)

Let  $\Gamma \vdash M, N : \forall X.\sigma$ . If  $M\tau =_{Fc} N\tau$  for some type  $\tau$ , then  $M =_{Fc} N$ .

Proof: Choose a fresh type variable  $Z$ .

Then,  $\Gamma \vdash MZ, NZ : [Z/X]\sigma$  and  $[\tau/Z](MZ) \equiv M\tau =_{Fc} N\tau \equiv [\tau/Z](NZ)$

Hence, applying the Main Lemma (Lemma 9.2),  $MZ =_{Fc} NZ$ .

Observe that  $Z$  fresh means  $Z$  not free in the type of any free term variable in  $MZ$  or  $NZ$ .

So, by  $\forall$ -intro,  $\lambda Z.MZ$  and  $\lambda Z.NZ$  are well-typed terms (of type  $\forall Z.[Z/X]\sigma$ ).

Hence, by  $\xi_2$ ,  $\lambda Z.MZ =_{Fc} \lambda Z.NZ$ , and, by  $\eta_2$ ,  $M =_{Fc} N$ . ■

## 10 Models

In this section, we outline the validity of Axiom C in some relevant models. Details and further references about the model theory of system F may be found in [AL91] or [Hyl]. The reader may also see [LM91] for an introductory presentation of PER models and [GLT89]

or [CGW88] for models based on coherent spaces or dI-domains. These constructions provide the main concrete paradigms for the general semantics of impredicative Type Theory and, by this, they allow a more explicit understanding of the semantic problems we will mention at the very end.

In short, in PER models, types are interpreted as partial equivalence relations (p.e.r.) on an arbitrary (partial) combinatory algebra  $(D, \cdot)$ , that is, on a model of (partial) Combinatory Logic. In other words, a type is a quotient of a subset of  $D$  modulo an equivalence relation. The terms of system F are interpreted as equivalence classes in these quotient sets. Given  $d \in D$ , call  $[d]_A$  the equivalence class of  $d$  in the p.e.r.  $A$ . Now,  $(D, \cdot)$  yields a model of the type free  $\lambda$ -calculus  $(D, \cdot, \llbracket - \rrbracket)$ , see [Bar84]. Set then  $er(M)$  for the term of system F with all types erased (e.g.,  $er(\lambda x : \tau. M \rho) = \lambda x. er(M)$ ) and consider  $\llbracket er(M) \rrbracket_\xi$ , i.e., the interpretation in  $D$ , under term environment  $\xi$ , of the type-free term  $er(M)$ . A result in [Mit86] (see also [CL91]) shows that the meaning in the PER model of a term  $M$  of system F is given by the equivalence class of the meaning of its erasure in the p.e.r. that interprets its type. More formally, if environment  $\xi'$  is obtained from  $\xi$  by forgetting type information,

$$\llbracket \Gamma \vdash M : \sigma \rrbracket_\xi = \llbracket \llbracket er(M) \rrbracket_{\xi'} \rrbracket_{\llbracket \sigma \rrbracket}$$

It is then clear that PER models realize Axiom C: if  $M\tau$  and  $M\tau'$  live in the same type  $\sigma$ , then their meanings are identical as  $er(M\tau) = er(M\tau')$ .

As for dI-based models, we recall here only that these may be constructed over the category of coherent spaces and stable maps, as in [Gir86], or over proper dI-domains as in [CGW88], which we follow. Types then are dI-domains or, more precisely, in view of possibly free type variables, they are maps over dI-domains. Indeed, they may be understood as functors if one considers the subcategory  $DI^L$  of dI-domains and just rigid embeddings as maps, as in [CGW88]. (The impossibility of viewing types as functors, in general, was discussed in the introduction, in view of the (contra-) and (co-)variance of the  $\rightarrow$  functor.) In short, let  $F : DI^L \rightarrow DI^L$  be a functor. Then  $\Pi F$ , the product functor meant to interpret impredicative second-order types, is simply the collection of uniform families  $(t_X)$ , where  $X$  ranges over dI-domains, such that  $t_X \in F(X)$  and  $t_X = F(f)^R t_Y$  for any dI-domain  $Y$  and any morphism  $f$  from  $X$  to  $Y$ . Assume now that  $\forall X. \sigma$  is such that  $X$  is not free in  $\sigma$ . This means that  $\sigma$  is interpreted by a constant functor  $F$ , with respect to  $X$ . Then  $F(f)^R = F(f) = id$  always. In particular, take  $Y$  as the universal domain, i.e., any other may be rigidly embedded in it. Then, for any uniform family  $(t_X)$  and any  $X$ , one has  $t_X = t_Y$  in  $F(X)$ . This is exactly the validity of Axiom C in these models.

There are several ways to describe the general (categorical) semantics of system F. In order to give a general meaning to Axiom C, we follow the presentation by internal categories given in [AL91]. First, though, the naïve, set-theoretic approach may guide our intuition. Let  $Tp$  be the collection of semantic types. A variable type is then a function  $F : Tp \rightarrow Tp$ . As usual, a product indexed over  $Tp$  is given by the set

$$\Pi F = \{f : Tp \rightarrow \cup F \mid \forall X \in Tp \ f(X) \in F(X)\}$$



Then Axiom C corresponds to

$$\text{if } f \in \Pi F \text{ and } \exists A \forall B F(B) = A, \text{ then } \exists a \in A \forall B f(B) = a$$

Or, also,  $\Pi F$  and  $A$  are set-theoretically isomorphic, when  $F$  is constantly equal to  $A$ . We know though that classical Set Theory does not yield models of impredicative Type Theory. However, models may be found as categories which are internal not to the category of sets and functions, but to more “constructive” ones, which enjoy the fundamental adjunction (Adj) below. Following [AL91], let  $c = (c_0, c_1)$  be a category internal to a Cartesian Closed Category (ccc)  $E$  with all finite limits. Let  $c^{c_0}$  be the category of internal functors. Then  $(E, c)$  yields a model of system F if  $c$  is an internal ccc and the (internal) product functor  $\Pi : c^{c_0} \rightarrow c$  exists as the right adjoint of the (internal) diagonal functor  $K : c \rightarrow c^{c_0}$ , i.e., the functor that to each  $A$  associates the functor  $K A$ , which is constant  $A$ . In other words,

$$(\text{Adj}) \quad c^{c_0}[K \_, \_] \cong c[\_, \Pi \_]$$

We claim that, among these models, exactly those which realize the following natural isomorphism

$$(\text{Const}) \quad c^{c_0}[K \_, K \_] \cong c[\_, \_]$$

are models of Axiom C. Indeed, by (Adj), (Const) implies, naturally in  $A, B$ ,

$$c[B, \Pi(K A)] \cong c^{c_0}[K B, K A] \cong c[B, A]$$

This is equivalent, in these models, to the isomorphism  $\Pi(K A) \cong A$ , i.e., to the intuitive set-theoretic meaning of Axiom C. A final remark: both the term model of system F, of course, and the retraction models (see [AL91]) do not realize Axiom C.

The semantics of the Genericity Theorem raises some interesting issues. Observe that

$$(\text{GEN}) \quad \exists \tau M\tau = N\tau \Rightarrow M = N$$

is not an equation, but an implication between equations. Thus, a model  $\mathcal{M}$  of Fc does not need to realize (GEN), in the sense that  $\exists \tau M\tau = N\tau$  may be true in the model but  $M = N$  is false. For example, PER models and dI-domains do not realize (GEN). Consider  $0, K : \forall X.X \rightarrow (X \rightarrow X)$ . Take then a type  $\tau$  which has at most one element, for instance  $\forall X.X$  or  $\forall X.X \rightarrow X$ . Then, in both classes of models,  $K\tau = 0\tau$ , but, of course,  $K \neq 0$ . By generalizing this argument (see [Lon93]), models of relational parametricity also do not realize (GEN). This lack (so far) of models of (GEN) is in spite of the many models of Fc and the provability of the implication. Note that an understanding of the semantics is relevant, not only for model-theoretic reasons, but also for the extensions of system F which are relevant in practice. That is, actual polymorphic functional languages may be based on core calculi, plus possibly more equation schemes. Thus, the investigation of which equational theories realize (GEN), as an important property of polymorphic functions, is a further challenge.

## References

- [ACC93] M. Abadi, L. Cardelli, and P.-L. Curien. Formal Parametric Polymorphism. In *Proceedings of the 20th Symposium on Principles of Programming Languages*, Charleston, South Carolina (January 1993).
- [AL91] A. Asperti and G. Longo. *Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist*. MIT Press (1991).
- [BFSS90] E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial Polymorphism. *Theoretical Computer Science*, 70:35–64 (January 1990). Corrigendum in 71:431 (April 1990).
- [Bar84] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics 103, North-Holland (1984). Revised edition.
- [BS93] H.P. Barendregt and R. Statman. *Typed Lambda Calculus with Applications*. In preparation.
- [CL91] L. Cardelli and G. Longo. A Semantic Basis for Quest. *Journal of Functional Programming*, 1:417–458 (April 1991).
- [CMMS91] L. Cardelli, S. Martini, J.C. Mitchell, and A. Scedrov. An Extension of System F with Subtyping. In *Proceedings of the Conference on Theoretical Aspects of Computer Software*, Sendai, Japan (September 1991). Lecture Notes in Computer Science 526, Springer-Verlag. Edited by T. Ito and R. Meyer.
- [CGL92] G. Castagna, G. Ghelli, and G. Longo. A Calculus for Overloaded Functions with Subtyping. In *Proceedings of the Conference on LISP and Functional Programming*, San Francisco, California (July 1992). Extended abstract.
- [CGW88] T. Coquand, C.A. Gunter, and G. Winskel. DI-Domains as a Model of Polymorphism. In *Proceedings of the 3rd Workshop on Mathematical Foundations of Programming Language Semantics*, New Orleans, Louisiana (April 1987). Lecture Notes in Computer Science 298, Springer-Verlag. Edited by M. Main, A. Melton, M. Mislove, and D. Schmidt.
- [DL89] R. Di Cosmo and G. Longo. Constructively Equivalent Propositions and Isomorphisms of Objects (or Terms as Natural Transformations). In *Proceedings of the Workshop on Logic for Computer Science*, Berkeley, California (November 1989). Mathematical Sciences Research Institute Publications 21, Springer-Verlag (1992). Edited by Y.N. Moschovakis.
- [EK66] S. Eilenberg and G.M. Kelly. A Generalization of the Functorial Calculus. *Journal of Algebra*, 3:366–375 (1966).

- [FGSS88] P.J. Freyd, J.-Y. Girard, A. Scedrov, and P.J. Scott. Semantic Parametricity in Polymorphic Lambda-Calculus. In *Proceedings of the 3rd Symposium on Logic in Computer Science*, Edinburgh, Scotland (June 1988).
- [Gir71] J.-Y. Girard. Une Extension de l'Interpretation Fonctionnelle de Gödel à l'Analyse et son Application à l'Elimination des Coupures dans l'Analyse et la Théorie des Types. In *Proceedings of the 2nd Scandinavian Logic Symposium*. North-Holland (1971). Edited by J.F. Fenstad.
- [Gir86] J.-Y. Girard. The System F of Variable Types, Fifteen Years Later. *Theoretical Computer Science*, 45:159–192 (1986).
- [GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press (1989).
- [GSS] J.-Y. Girard, A. Scedrov, and P.J. Scott. Normal Forms and Cut-Free Proofs as Natural Transformations. In *Proceedings of the Workshop on Logic for Computer Science*, Berkeley, California (November 1989). Mathematical Sciences Research Institute Publications 21, Springer-Verlag (1992). Edited by Y.N. Moschovakis.
- [Has93] R. Hasegawa. Categorical Data Types in Parametric Polymorphism. To appear in *Mathematical Structures in Computer Science*.
- [Hyl] M. Hyland. A Small Complete Category. *Annals of Pure and Applied Logic*, 40:135–165 (1988).
- [LS86] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics 7, Cambridge University Press (1986).
- [Lon93] G. Longo. Parametric and Type-Dependent Polymorphism. To appear in *Fundamenta Informatica*.
- [LM91] G. Longo and E. Moggi. Constructive Natural Deduction and its  $\omega$ -set Interpretation. *Mathematical Structures in Computer Science*, 1(2):215–253 (1991).
- [MR91] Q. Ma and J.C. Reynolds. Types, Abstraction, and Parametric Polymorphism: Part 2. In *Proceedings of the Workshop on Mathematical Foundations of Programming Language Semantics*. Lecture Notes in Computer Science, Springer-Verlag (1991). Edited by S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt.
- [Mac71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag (1971).
- [Mit86] J.C. Mitchell. A Type Inference Approach to Reduction Properties and Semantics of Polymorphic Expressions. In *Proceedings of the Conference on LISP and Functional Programming* (1986).

- [PA93] G. Plotkin and M. Abadi. A Logic for Parametric Polymorphism. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, Utrecht, Netherlands (March 1993). Lecture Notes in Computer Science 664, Springer-Verlag (1993). Edited by M. Bezem and J.F. Groote.
- [Rey74] J.C. Reynolds. Towards a Theory of Type Structure. In *Proceedings of le Colloque sur la Programmation*. Lecture Notes in Computer Science 19, Springer-Verlag (1974). Edited by B. Robinet.
- [Rey83] J.C. Reynolds. Types, Abstraction and Parametric Polymorphism. *Information Processing*, 83:513–523 (1983). North-Holland. Edited by R.E.A. Mason.
- [Sco72] D. Scott. Continuous lattices. *Toposes, Algebraic Geometry and Logic*. Lecture Notes in Mathematics 274, Springer-Verlag (1972). Edited by F.W. Lawvere.
- [SP82] M. Smyth and G. Plotkin. The Category Theoretic Solution of Recursive Domain Equations. *SIAM Journal of Computing*, 11:761–783 (1982).
- [Str67] C. Strachey. *Fundamental Concepts in Programming Languages*. Unpublished lecture notes from the International Summer School in Computer Programming, Copenhagen, Denmark (August 1967).

## PRL Research Reports

The following documents may be ordered by regular mail from:

Librarian – Research Reports  
Digital Equipment Corporation  
Paris Research Laboratory  
85, avenue Victor Hugo  
92563 Rueil-Malmaison Cedex  
France.

It is also possible to obtain them by electronic mail. For more information, send a message whose subject line is `help to doc-server@prl.dec.com` or, from within Digital, to `decprl : doc-server`.

Research Report 1: *Incremental Computation of Planar Maps*. Michel Gangnet, Jean-Claude Hervé, Thierry Pudet, and Jean-Manuel Van Thong. May 1989.

Research Report 2: *BigNum: A Portable and Efficient Package for Arbitrary-Precision Arithmetic*. Bernard Serpette, Jean Vuillemin, and Jean-Claude Hervé. May 1989.

Research Report 3: *Introduction to Programmable Active Memories*. Patrice Bertin, Didier Roncin, and Jean Vuillemin. June 1989.

Research Report 4: *Compiling Pattern Matching by Term Decomposition*. Laurence Puel and Ascánder Suárez. January 1990.

Research Report 5: *The WAM: A (Real) Tutorial*. Hassan Aït-Kaci. January 1990.<sup>†</sup>

Research Report 6: *Binary Periodic Synchronizing Sequences*. Marcin Skubiszewski. May 1991.

Research Report 7: *The Siphon: Managing Distant Replicated Repositories*. Francis J. Prusker and Edward P. Wobber. May 1991.

Research Report 8: *Constructive Logics. Part I: A Tutorial on Proof Systems and Typed  $\lambda$ -Calculi*. Jean Gallier. May 1991.

Research Report 9: *Constructive Logics. Part II: Linear Logic and Proof Nets*. Jean Gallier. May 1991.

Research Report 10: *Pattern Matching in Order-Sorted Languages*. Delia Kesner. May 1991.

---

<sup>†</sup>This report is no longer available from PRL. A revised version has now appeared as a book: “Hassan Aït-Kaci, *Warren’s Abstract Machine: A Tutorial Reconstruction*. MIT Press, Cambridge, MA (1991).”

Research Report 11: *Towards a Meaning of LIFE*. Hassan Aït-Kaci and Andreas Podelski. June 1991 (Revised, October 1992).

Research Report 12: *Residuation and Guarded Rules for Constraint Logic Programming*. Gert Smolka. June 1991.

Research Report 13: *Functions as Passive Constraints in LIFE*. Hassan Aït-Kaci and Andreas Podelski. June 1991 (Revised, November 1992).

Research Report 14: *Automatic Motion Planning for Complex Articulated Bodies*. Jérôme Barraquand. June 1991.

Research Report 15: *A Hardware Implementation of Pure Esterel*. Gérard Berry. July 1991.

Research Report 16: *Contribution à la Résolution Numérique des Équations de Laplace et de la Chaleur*. Jean Vuillemin. February 1992.

Research Report 17: *Inferring Graphical Constraints with Rockit*. Solange Karsenty, James A. Landay, and Chris Weikart. March 1992.

Research Report 18: *Abstract Interpretation by Dynamic Partitioning*. François Bourdoncle. March 1992.

Research Report 19: *Measuring System Performance with Reprogrammable Hardware*. Mark Shand. August 1992.

Research Report 20: *A Feature Constraint System for Logic Programming with Entailment*. Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. November 1992.

Research Report 21: *The Genericity Theorem and the Notion of Parametricity in the Polymorphic  $\lambda$ -calculus*. Giuseppe Longo, Kathleen Milsted, and Sergei Soloviev. December 1992.

Research Report 22: *Sémantiques des langages impératifs d'ordre supérieur et interprétation abstraite*. François Bourdoncle. January 1993.

Research Report 23: *Dessin à main levée et courbes de Bézier : comparaison des algorithmes de subdivision, modélisation des épaisseurs variables*. Thierry Pudet. January 1993.

Research Report 24: *Programmable Active Memories: a Performance Assessment*. Patrice Bertin, Didier Roncin, and Jean Vuillemin. March 1993.

Research Report 25: *On Circuits and Numbers*. Jean Vuillemin. April 1993.

Research Report 26: *Numerical Valuation of High Dimensional Multivariate European Securities*. Jérôme Barraquand. March 1993.

Research Report 27: *A Database Interface for Complex Objects*. Marcel Holsheimer, Rolf A. de By, and Hassan Aït-Kaci. March 1993.

Research Report 28: *Feature Automata and Sets of Feature Trees*. Joachim Niehren and Andreas Podelski. March 1993.

Research Report 29: *Real Time Fitting of Pressure Brushstrokes*. Thierry Pudet. March 1993.

Research Report 30: *Rollit: An Application Builder*. Solange Karsenty and Chris Weikart. April 1993.

Research Report 31: *Label-Selective  $\lambda$ -Calculus*. Hassan Aït-Kaci and Jacques Garrigue. May 1993.

Research Report 32: *Order-Sorted Feature Theory Unification*. Hassan Aït-Kaci, Andreas Podelski, and Seth Copen Goldstein. May 1993.

