The Genericity Theorem and the Notion of Parametricity in the Polymorphic λ-calculus

Giuseppe LONGO Kathleen MILSTED*
Sergei SOLOVIEV**

Laboratoire d'Informatique, URA 1327 du CNRS
Département de Mathématiques et d'Informatique
Ecole Normale Supérieure
*Digital Equipment Corporation, Paris
**Russian Academy of Sciences, St. Petersburg

LIENS - 92 - 25

December 1992

The Genericity Theorem and the Notion of Parametricity in the Polymorphic λ-calculus

Giuseppe Longo[1] Kathleen Milsted[2] Sergei Soloviev[3]

December 1992

Abstract

This paper focuses on how terms of second order λ-calculus, which may take types as inputs, depend on types. These terms are generally understood to have an "essentially" constant meaning, in all models, on input types. We show how the proof theory of second order λ-calculus suggests a clear syntactic description of this phenomenon. Namely, under a reasonable condition, we show that identity of two polymorphic functions on a single type implies identity of the functions (equivalently, every type is a generic input).

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 λ-calculus notation, where capital $X, Y, \ldots$ stand for type variables, one is allowed to consider terms such as $\lambda X.M$ which may be fed a type as input and give a term as output (in the logical jargon, $\lambda X.M$ is a second order term in impredicative Type Theory).

Since early remarks of Strachey [Str67], a distinction was introduced on how these explicitly polymorphic functions had to 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

[1]LIENS(CNRS)-DMI, Ecole Normale Supérieure, 45 rue d'Ulm, 75005 Paris, France. *longo@dmi.ens.fr*

[2]Digital Equipment Corporation, Paris Research Laboratory (PRL), 85 avenue Victor Hugo, 92563 Rueil-Malmaison, France. *milsted@prl.dec.com*

[3]*Currently at:* Computer Science Department, Aarhus University, Ny Munkegade, Bd. 540, DK 8000, Aarhus C, Denmark. *soloviev@daimi.aau.dk*

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).

Following Strachey (and Reynolds) then, "proper" polymorphism, as opposed to the "ad hoc" variety, was understood as a property of second order terms to define functions that have a "uniform" dependency on input types, or whose output terms do not "essentially" depend on input types. Note, though, that the output terms of, say, $\lambda X.M$ applied to types σ and τ, 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 λ-calculus), satisfy this uniformity property or essential independence from input types 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 σ and τ, then (the interpretation of) $\lambda X.M$, applied to (the meaning of) σ and τ, should send related elements of σ and τ to related elements in the types of the outputs.

Another meaning of the proper polymorphism of system F was given 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 provide the core way to express uniformity on objects (as interpretation of types) in categories. Unfortunately, natural transformations act on functors and, in general categories, variable types are not functors. The counterexample is straightforward: the map from X to $X \to X$ (the arrow type) should be at once a covariant and contravariant functor. A partial solution, in the context of the typed λ-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 fine (i.e., it is compositional) on relevant models (see [BFSS90, FGSS88, GSS]). On essentially similar lines, Freyd suggested the novel notion of "structor" in order to understand, categorically, the notion of uniformity inherent in second order λ-terms.

These attempts suggested brand new constructions and relevant mathematics, but seem still insufficient to fill the essential gap between the so-called parametricity or uniformity of second order λ-calculus and the core 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].

We propose in this work a simple "syntactic" understanding of the so-called parametricity property, or "uniform dependency" on input types, of polymorphic calculi. First, we present an extension of system F, suggested by a simple result of Girard. In [Gir71], given a type σ and a term J_σ such that, for any type τ, $J_\sigma \tau$ reduces to 1 if $\sigma = \tau$, and reduces to 0 if $\sigma \neq \tau$, then $\mathrm{F} + J_\sigma$ does not normalize. Since system F normalizes, J_σ is not definable in F. The point here is that the polymorphic term J_σ gives essentially different output terms living in the same type, according to the (values of the) input types. Then, the starting point for 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 axiom:

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

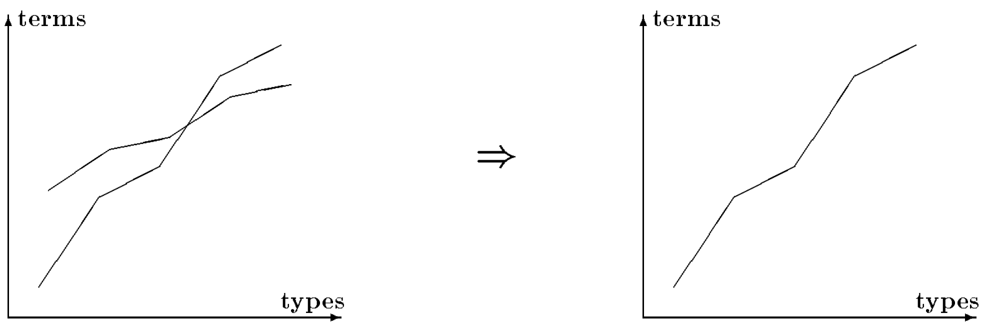
That is, if the outputs of a polymorphic term M, applied to any type, all live in the same type σ, then these outputs are simply equal. Axiom C is not provable in F, but it is compatible with F, i.e., F may be consistently extended with it. Indeed, a generalization of Axiom C appears in the system $\mathrm{F}_{<:}$ [CMMS91] which extends system F with subtyping; see rule *Eq appl2*. Moreover, in section 10, we point out that both the PER models in realizability topoi and Girard's models over dI-domains and stable maps satisfy Axiom C. A categorical characterization of models realizing Axiom C will also be outlined. In our view, this obvious remark, the compatibility of Axiom C with system F, is one thing to be noted in order to understand parametricity. From ongoing work [ACC92], it also turns out that Axiom C is realized by all models that satisfy Reynolds's parametricity condition [MR91].

Consider now Fc, the extension of system F with Axiom C. What we propose for a syntactic understanding of parametricity 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 τ, then $M =_{Fc} N$

The reader should notice where intended parentheses and existential quantification are located, and also, that there is no restriction on σ. The Genericity Theorem says that, in Fc, if two second order terms coincide on one type, they coincide everywhere. Or, equivalently, that each single type acts as a "generic" input, as a variable.

As second order terms are functions from types to terms, the following intuitive geometric description of our understanding of parametricity may further help the reader:



This illustrates that if two polymorphic functions coincide on an input type, then they are, in fact, the same function. In a sense, there are "very few" polymorphic functions.

Of course, more things must be settled, due to the fact that the Genericity Theorem is stated in Fc, not in F. However, Axiom C is a very natural request, directly derived from the old and simple result $J_\sigma \tau$ in [Gir71]. Its relevance is also confirmed by the use made of it in [CMMS91]. We will discuss the model theoretic problems raised by the Genericity Theorem in section 10. The next sections recall system F, introduce system Fc, 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:

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

We will use σ, τ, ρ, μ, ν 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, $\to$ associates to the right, and the scope of $\forall$ and λ 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 substitution of a free (type or term) variable is also defined as usual, and written $[\tau/X]P$ or $[M/x]P$.

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. In general, we will use Γ to denote a set of declarations, and we write $\Gamma, x : \sigma$ to extend Γ with a new declaration

$x : \sigma$, where x must not occur in Γ. A *type assignment* is a meta-expression of the form $\Gamma \vdash M : \sigma$, which asserts that term M has, or lives in, type σ, relative to the declarations in Γ. The following rules define valid type assignments.

Type Assignment Rules

(declaration) $\quad \Gamma, x{:}\sigma \vdash x : \sigma$

($\rightarrow$-intro) $$\frac{\Gamma, x{:}\sigma \vdash M : \tau}{\Gamma \vdash \lambda x{:}\sigma.M : \sigma \rightarrow \tau}$$

($\rightarrow$-elim) $$\frac{\Gamma \vdash M : \sigma \rightarrow \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

*($\forall$-intro) $$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \lambda X.M : \forall X.\sigma}$$

* for X not free in the type of any free term variable in M

($\forall$-elim) $$\frac{\Gamma \vdash M : \forall X.\sigma}{\Gamma \vdash M\tau : [\tau/X]\sigma}$$

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.

Equal terms are defined by the following schemes and rules:

Equational Schemes and Rules

(β_1) $\quad (\lambda x{:}\sigma.M)N \;=\; [N/x]M$ $\qquad$ (β_2) $\quad (\lambda X.M)\tau \;=\; [\tau/X]M$

(η_1) $\quad \lambda x{:}\sigma.Mx \;=\; M$ for $x \notin FV(M)$ $\qquad$ (η_2) $\quad \lambda X.MX \;=\; M$ for $X \notin FV(M)$

(ξ_1) $$\frac{M = N}{\lambda x{:}\sigma.M = \lambda x{:}\sigma.N}$$

(ξ_2) $$\frac{M = N}{\lambda X.M = \lambda X.N}$$

(app$_1$) $$\frac{M_1 = M_2 \quad N_1 = N_2}{M_1N_1 = M_2N_2}$$

(app$_2$) $$\frac{M = N}{M\tau = N\tau}$$

(refl) $\quad M = M$

(sym) $$\frac{M_1 = M_2}{M_2 = M_1}$$

(trans) $$\frac{M_1 = M_2 \quad M_2 = M_3}{M_1 = M_3}$$

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:

(β_1) $\quad (\lambda x{:}\sigma.M)N \longrightarrow_{\beta_1} [N/x]M$ $\qquad$ (β_2) $\quad (\lambda X.M)\tau \longrightarrow_{\beta_2} [\tau/X]M$
(η_1) $\quad \lambda x{:}\sigma.Mx \longrightarrow_{\eta_1} M$ for $x \notin FV(M)$ $\qquad$ (η_2) $\quad \lambda X.MX \longrightarrow_{\eta_2} M$ for $X \notin FV(M)$

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

Equational Church-Rosser
If $M_1 = M_2$ *then there exists an* M_0 *such that* $M_1 \longrightarrow_F M_0$ *and* $M_2 \longrightarrow_F M_0$.

3 System Fc

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

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

If the outputs of polymorphic function M live in a type σ 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 authorizes more equations 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, strong normalization, and the Church-Rosser property. 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:

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

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.*

Case: 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 β_2, $[\tau/X]M =_{\beta_2} (\lambda X.M)\tau =_c (\lambda X.M)\tau' =_{\beta_2} [\tau'/X]M$.
Case: Axiom C* implies Axiom C.
Assume that $\Gamma \vdash M : \forall X.\sigma$ and $X \notin FV(\sigma)$.
Let Z be a fresh type variable. Then, $\Gamma \vdash MZ : \sigma$, and Z not free in any of Γ, M, σ.
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 τ, then $M =_{Fc} N$

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

Assume M and N live in the same type σ
If $[\tau/X]M =_{Fc} [\tau/X]N$ for some type τ, 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 this.

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, apply Axiom C* to $zZX(xZ) : X$ to obtain

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

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

However, this example also provides a hint to the proof of the Genericity Theorem. 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 λ-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).

The proof proceeds next with a property of C*-equality that we call *Quasi-Genericity*: if a term has a type substitution structure, i.e., is of the form $[\tau/X]M$, and Axiom C* is applied to it, then that exact type substitution structure is preserved, i.e., the result is of the form $[\tau/X]N$, and, moreover, $M =_{c^*} N$. This is shown in section 6, where we also give a counter-example to show that F-equality does **not** satisfy this property.

One of the consequences of Quasi-Genericity of C*-equality is that if the result of applying Axiom C* to $[\tau/X]M$ is already known to be of the form $[\tau/X]N$, then $M =_{c^*} N$. Observe that this is a weak form of Genericity, in that it uses C*-equality instead of Fc-equality in the premise of the Genericity Theorem. Another weak form, with F-equality in the premise, can also be proven. Thus, in section 7, we have the following two Weak Genericity Theorems:

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

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

Putting these two together to obtain Fc-equality in the premise depends on how C*-equality interacts with F-equality. Unfortunately, F and C*-equalities *cannot* be arbitrarily reordered. In section 8, we show that C*-equality and forward $\beta_1\beta_2\eta_1$ reduction (but not η_2 reduction) do indeed commute. In section 9, we draw all the pieces together to prove the main result. This involves examining the chain of Fc-equalities between $[\tau/X]M =_{Fc} [\tau/X]N$. Using the Church-Rosser property, commutativity of C*-equality with $\beta_1\beta_2\eta_1$ reduction, 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 ends up with $M =_{Fc} N$. This gives the result.

5 Type and Term Generalizers

In this section, we give a notion of "generalizer" and construct some critical ones for types and terms. In short, a generalizer of two types or terms may be instantiated to those types or terms, under suitable conditions.

For example, given two terms M_1 and M_2, if one has $[\tau/X]M_1 \equiv [\rho/Y]M_2$, then an abstract notion of a generalizer, with respect to a fresh type variable Z, will be a term M_0 such that, for suitable types μ_1, μ_2, one has

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

That is, if two terms are unified as above, then we will construct a common "term schema" which is instantiated, by type substitutions, to both of them. We use generalizers in later sections, where we show that the typing of M_0 permits Axiom C* to be applied, thus allowing us to deduce that $M_1 =_{Fc} M_2$.

The generalizers, though, that we construct here require more details, starting with an analysis of mutual occurrences of τ in ρ or ρ in τ.

Definition: in_k
If there are $k \geq 0$ occurrences of type τ in type ρ, we will write $\tau\ in_k\ \rho$.

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

If $\tau\ in_k\ \rho$ with $k \geq 0$, then, given fresh X, there are 2^k different X-contexts for τ in ρ. We will assume given an enumeration of these contexts, which we will write as $\rho_1^X, \ldots, \rho_h^X$ where $h = 2^k$. By convention, we take ρ_1^X to be ρ. For example, if $\tau = \rho$, then there are two X-contexts for τ in ρ, i.e., $\rho_1^X = \rho$ 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 τ and ρ, then we will assume that X and Y are not free in τ and ρ.

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 τ and ρ.

- *Case: $\tau\ in_k\ \rho$ for $k > 0$.*
  *Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, we say that P_0 is a $Z_0, \ldots, Z_h$-**generalizer** of P_1 and P_2 iff X and Y are not free in P_0 and*

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

  where $\rho_1^X, \ldots, \rho_h^X$ are the X-contexts for τ in ρ.

- *Case: ρ in_k τ for $k \geq 0$ and the previous case does not apply.*
  Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, we say that P_0 is a $Z_0, \ldots, Z_h$- ***generalizer*** *of P_1 and P_2 iff X and Y are not free in P_0 and*

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

  where $\tau_1^Y, \ldots, \tau_h^Y$ are the Y-contexts for ρ in τ.

Remark that, if $\tau = \rho$, then the first case of the definition applies, by τ in_1 ρ, giving

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

If τ and ρ are unrelated (i.e., they do not occur in each other), then the second case applies, by ρ in_0 τ:

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

As written, the cases are exclusive, for example, one cannot have both τ in_0 ρ and ρ in_0 τ, nor both ρ in_0 τ and τ in_k ρ, which would otherwise be possible.

Lemma 5.1 (Type Generalization)
Let σ_1, σ_2 be two types such that $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$ for some types τ and ρ. Assume that k is given either by τ in_k ρ for $k > 0$, or ρ in_k τ for $k \geq 0$ and the previous case does not apply. Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, there exists a type σ_0 that is a $Z_0, \ldots, Z_h$-generalizer of σ_1 and σ_2.

Proof: Assume first that τ in_k ρ for $k > 0$, and let $\sigma = [\tau/X]\sigma_1 = [\rho/Y]\sigma_2$.
τ-marking: Mark in σ those occurrences of τ which derive from σ_1 by a $[\tau/X]$ substitution.
σ-marking: Mark in σ those occurrences of ρ which derive from σ_2 by a $[\rho/Y]$ substitution.
Observe that some of the marked τs may appear in a marked ρ.
Construct then σ_0 from σ by the following procedure:
1) Replace by Z_0 all marked τs that do not occur in a marked ρ.
2) Consider now a marked ρ, possibly containing marked τs.
   Let ρ_i^X be the corresponding X-context in ρ for the marked τs. (If there are no marked τs, this will be $\rho_1^X \equiv \rho$). Replace the marked ρ by Z_i.

In case ρ in_k τ for $k \geq 0$ and the previous case does not apply, perform the τ and σ-markings as above and observe that some of the marked ρs may appear in a marked τ. Then, apply the dual procedure, where the roles of ρ and τ in steps 1) and 2) are interchanged (and τ_i^Y, the Y-contexts for ρ in τ, are used instead of ρ_i^X, the X-contexts for τ in ρ).
By construction, σ_0 satisfies the definition of a generalizer. ∎

In the following lemma, we show that, once fresh variables $Z_0, \ldots, Z_h$ 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 σ_1, σ_2 be two types such that $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$ for some types τ and ρ. Assume that k is given either by τ in_k ρ for $k > 0$, or ρ in_k τ for $k \geq 0$ and the previous case does not apply. Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, the $Z_0, \ldots, Z_h$-generalizer of σ_1 and σ_2 is unique.

Proof: Assume first that τ in_k ρ for $k > 0$.
Let σ_0 and σ_0' be two $Z_0, \ldots, Z_h$-generalizers of σ_1, σ_2. Then, by definition,

$$\begin{array}{rclclr} [\,X/Z_0,\, \rho_1^X/Z_1,\, \ldots,\, \rho_h^X/Z_h\,]\,\sigma_0 & = & \sigma_1 & = & [\,X/Z_0,\, \rho_1^X/Z_1,\, \ldots,\, \rho_h^X/Z_h\,]\,\sigma_0' & (1) \\ [\,\tau/Z_0,\;\; Y/Z_1,\, \ldots,\;\; Y/Z_h\,]\,\sigma_0 & = & \sigma_2 & = & [\,\tau/Z_0,\;\; Y/Z_1,\, \ldots,\;\; Y/Z_h\,]\,\sigma_0' & (2) \end{array}$$

with X and Y not free in σ_0 or σ_0'. We will show that $\sigma_0 = \sigma_0'$ by induction on σ_0.

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

$$X = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\, \sigma_0'$$
$$\tau = \sigma_2 = [\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\, \sigma_0'$$

We now consider the possible choices for σ_0'. Clearly, σ_0' cannot be X since $X \notin FV(\sigma_0')$. Nor can σ_0' be τ since then, (1) becomes $X = \sigma_1 = \tau$ but, by the substitution convention, $X \notin FV(\tau)$. Further, σ_0' cannot be Z_i for some $i = 1 \ldots h$, 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 $i = 1 \ldots h$. Then, (1) and (2) become

$$\rho_i^X = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\, \sigma_0'$$
$$Y = \sigma_2 = [\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\, \sigma_0'$$

First, σ_0' cannot be Y since $Y \notin FV(\sigma_0')$. Furthermore, σ_0' cannot be ρ_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 $i = 2 \ldots h$, $X \in FV(\rho_i^X)$ but $X \notin FV(\sigma_0')$. Also, σ_0' cannot be Z_0 since then, (2) becomes $Y = \sigma_2 = \tau$ but, by the substitution convention again, $Y \notin FV(\tau)$. Similarly, σ_0' cannot be Z_j for some $j = 1 \ldots h$ 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 $i = 0 \ldots h$. Then, (1) and (2) become

$$Z = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\, \sigma_0'$$
$$Z = \sigma_2 = [\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\, \sigma_0'$$

Since X and Y are not free in σ_0, then $Z \neq X$ and $Z \neq Y$ and, moreover, σ_0' cannot be Z_i for any $i = 0 \ldots h$. The only choice is $\sigma_0' \equiv Z = \sigma_0$.

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

$$[X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\,(\sigma \to \mu) = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\, \sigma_0'$$
$$[\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\,(\sigma \to \mu) = \sigma_2 = [\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\, \sigma_0'$$

Remark that σ_0' cannot be Z_i for any $i = 0 \ldots h$ since, then, a $\to$ 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, σ_0' must be of the form $\sigma' \to \mu'$, with σ, σ' and μ, μ' satisfying equations similar to (1) and (2). By induction, $\sigma = \sigma'$ and $\mu = \mu'$. Hence, $\sigma_0' \equiv \sigma' \to \mu' = \sigma \to \mu = \sigma_0$.

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

$$[X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\,(\forall Z.\sigma') = \sigma_1 = [X/Z_0, \rho_1^X/Z_1, \ldots, \rho_h^X/Z_h]\, \sigma_0'$$
$$[\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\,(\forall Z.\sigma') = \sigma_2 = [\tau/Z_0, Y/Z_1, \ldots, Y/Z_h]\, \sigma_0'$$

As with the previous case, σ_0' cannot be Z_i for any $i = 0 \ldots h$. So, σ_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 $\rho\ in_k\ \tau$ for $k \geq 0$ and not the previous case. ■

Fact 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 $\tau\ in_k\ \rho$ for $k > 0$, or $\rho\ in_k\ \tau$ for $k \geq 0$ and the previous case does not apply. Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, let σ_0 and μ_0 be the $Z_0, \ldots, Z_h$-generalizers of σ_1, σ_2 and μ_1, μ_2, respectively. Then, $[\mu_0/Z]\sigma_0$ is the $Z_0, \ldots, Z_h$-generalizer of $[\mu_1/Z]\sigma_1$ and $[\mu_2/Z]\sigma_2$.

Lemma 5.4 (Generalization of Declarations)
Let Γ_1, Γ_2 be two sets of declarations such that $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$. Assume that k is given either by τ in_k ρ for $k > 0$, or ρ in_k τ for $k \geq 0$ and the previous case does not apply. Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, there exists a set of declarations Γ_0 that is a unique $Z_0, \ldots, Z_h$-generalizer of Γ_1 and Γ_2.

Proof: Since $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$, then Γ_1 and Γ_2 must contain declarations of the same term variables. Thus, $\Gamma_1 \equiv x_1\!:\!\sigma_1^1, \ldots, x_n\!:\!\sigma_n^1$ and $\Gamma_2 \equiv x_1\!:\!\sigma_1^2, \ldots, x_n\!:\!\sigma_n^2$ with $[\tau/X]\sigma_i^1 = [\rho/Y]\sigma_i^2$ for $i = 1 \ldots n$.
For $i = 1 \ldots n$, construct the unique $Z_0, \ldots, Z_h$-generalizer σ_i^0 of σ_i^1 and σ_i^2.
Then, $\Gamma_0 \equiv x_1\!:\!\sigma_1^0, \ldots, x_n\!:\!\sigma_n^0$ is the unique $Z_0, \ldots, Z_h$-generalizer of Γ_1 and Γ_2. ■

The following 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 here 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 τ and ρ. Assume that k is given either by τ in_k ρ for $k > 0$, or ρ in_k τ for $k \geq 0$ and the previous case does not apply. Let $h = 2^k$. Given fresh type variables $Z_0, \ldots, Z_h$, there exist a set of declarations Γ_0, a term M_0, and a type σ_0 that are unique $Z_0, \ldots, Z_h$-generalizers of Γ_1, Γ_2; M_1, M_2; and σ_1, σ_2, respectively, such that $\Gamma_0 \vdash M_0 : \sigma_0$.

Proof: We will construct Γ_0, M_0, σ_0 by induction on the derivation of $\Gamma_1 \vdash M_1 : \sigma_1$. (In the proof, we will write simply "generalizer" instead of "$Z_0, \ldots, Z_h$-generalizer").
<u>*Case:*</u> Assume that $\Gamma_1 \vdash M_1 : \sigma_1$ by a variable declaration in Γ_1.
Then, $M_1 \equiv x$ and $x\!:\!\sigma_1 \in \Gamma_1$ for some term variable x.
¿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 $x\!:\!\sigma_2 \in \Gamma_2$.
And, since $[\tau/X]\Gamma_1 = [\rho/Y]\Gamma_2$ with $x\!:\!\sigma_1 \in \Gamma_1$ and $x\!:\!\sigma_2 \in \Gamma_2$, then $[\tau/X]\sigma_1 = [\rho/Y]\sigma_2$.
Take σ_0 to be the unique generalizer of σ_1, σ_2 by Type Generalization (Lemma 5.1), and Γ_0 to be the unique generalizer of Γ_1, Γ_2 by Lemma 5.4.
Observe that, by construction, $x : \sigma_0 \in \Gamma_0$, from which, by the declaration rule, $\Gamma_0 \vdash x : \sigma_0$. Since x is clearly the only generalizer of M_1, M_2, 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: Γ_0' of $(\Gamma_1, x\!:\!\mu_1), (\Gamma_2, x\!:\!\mu_2)$;
M_0' of M_1', M_2'; and ρ_0 of ρ_1, ρ_2, such that $\Gamma_0' \vdash M_0' : \rho_0$.
But, since generalizers of types and sets of declarations are unique, then Γ_0' must be $\Gamma_0, x\!:\!\mu_0$ where Γ_0 and μ_0 are unique generalizers of Γ_1, Γ_2 and μ_1, μ_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 σ_1, σ_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, μ_0' and M_0'', of μ_1, μ_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: Γ_0 of Γ_1, Γ_2; N_0' of N_1', N_2'; and ρ_0 of ρ_1, ρ_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 ρ' 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, ρ' must be $\rho_0 \rightarrow \sigma_0$, where ρ_0 and σ_0 are unique generalizers of ρ_1, ρ_2 and σ_1, σ_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'.
¿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: Γ_0 of Γ_1, Γ_2; M_0' of M_1', M_2'; and μ_0 of μ_1, μ_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 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 σ_1, σ_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 $[\tau/X]\mu_1 = [\rho/Y]\mu_2$.

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: Γ_0 of Γ_1, Γ_2; M_0' of M_1', M_2'; and ρ' of $\forall Z.\rho_1, \forall Z.\rho_2$, such that $\Gamma_0 \vdash M_0' : \rho'$.

Since type generalizers are unique, ρ' must be $\forall Z.\rho_0$, where ρ_0 is the generalizer of ρ_1, ρ_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 μ_0 is the unique generalizer of μ_1, μ_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 Fact 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 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 also of the form $[\tau/X]N$, and furthermore, $M =_{c^*} N$. We call this property *Quasi-Genericity* of C*-equality (since it resembles genericity). 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 6.1 (Quasi-Genericity of C*-equality)
If $[\tau/X]M =_{c^*} N'$ then there exists a term N such that $N' \equiv [\tau/X]N$ and $M =_{c^*} N$.

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

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

We prove only the case $[\tau/X]M \stackrel{1}{=}_{c^*} N'$ as the inductive case holds by transitivity.

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

Assume then that $[\tau/X]M \stackrel{1}{=}_{c^*} N'$.
Then, as previously remarked, 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, the proof proceeds by straightforward induction on the structure of M.
We display here the case when Axiom C* is applied to $[\tau/X]M$ itself (this includes the base case of the previous induction on M).
Then, by definition of Axiom C*, there exists a term M', types ρ, ρ', and a type variable Y, such that

$$[\tau/X]M \;\equiv\; [\rho/Y]M' \;\stackrel{1}{=}_{c^*}\; [\rho'/Y]M' \;\equiv\; N' \tag{3}$$

where, for $\Gamma \vdash M : \sigma$, we have $\Gamma \vdash M' : \sigma'$, and Y not free in Γ nor σ'.
Since Axiom C* is actually applied, then $Y \in FV(M')$ and, thus, $X \notin FV(\rho')$; otherwise X would be free in N', contradicting the assumption.
We now apply Term Generalization to $[\tau/X]M \equiv [\rho/Y]M'$.

Case: Assume that $\tau\ in_k\ \rho$ for $k > 0$.
Choose fresh type variables $Z_0, \ldots, Z_h$ where $h = 2^k$.
Then, by Term Generalization (Theorem 5.5), there exist unique $Z_0, \ldots, Z_h$-generalizers: Γ_0 of Γ, Γ; M_0 of M, M'; and σ_0 of σ, σ', such that $\Gamma_0 \vdash M_0 : \sigma_0$.
Observe now that, by definition of generalizer, we have
that $\Gamma = [\tau/Z_0,\ Y/Z_1,\ \ldots,\ Y/Z_h]\,\Gamma_0$ and $\sigma' = [\tau/Z_0,\ Y/Z_1,\ \ldots,\ Y/Z_h]\,\sigma_0$.
But, we also have that Y is not free in Γ or σ'.
So $Z_1, \ldots, Z_h$ cannot be free in Γ_0 or σ_0. Hence, since $\Gamma_0 \vdash M_0 : \sigma_0$, we can apply Axiom C* to M_0 for the variables $Z_1, \ldots, Z_h$.
Thus, if we take

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

we get the desired result, as

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

and

$$\begin{array}{rll} [\tau/X]\,N \equiv & [\tau/Z_0,\ \rho'/Z_1,\ \ldots,\ \rho'/Z_h]\,M_0 & \text{since } X \notin FV(\rho') \\ \equiv & [\rho'/Y]\,[\tau/Z_0,\ Y/Z_1,\ \ldots,\ Y/Z_h]\,M_0 & \text{by rearranging substitutions} \\ \equiv & [\rho'/Y]\,M' & M_0 \text{ is the generalizer of } M, M' \\ \equiv & N' & \text{by (3)} \end{array}$$

Case: Assume that $\rho\ in_k\ \tau$ for $k \geq 0$ and the previous case does not apply.
Choose fresh type variables $Z_0, \ldots, Z_h$ where $h = 2^k$.
Then, by Term Generalization (Theorem 5.5), there exist unique $Z_0, \ldots, Z_h$-generalizers: Γ_0 of Γ, Γ; M_0 of M, M'; and σ_0 of σ, σ', such that $\Gamma_0 \vdash M_0 : \sigma_0$.
Observe now that, by definition of generalizer, we have
that $\Gamma = [Y/Z_0,\ \tau/Z_1,\ \tau_2^Y/Z_2,\ \ldots,\ \tau_h^Y/Z_h]\,\Gamma_0$
and $\sigma' = [Y/Z_0,\ \tau/Z_1,\ \tau_2^Y/Z_2,\ \ldots,\ \tau_h^Y/Z_h]\,\sigma_0$.
But, we also have that Y is not free in Γ or σ'.
So $Z_0, Z_2, \ldots, Z_h$ cannot be free in Γ_0 or σ_0.
Hence, since $\Gamma_0 \vdash M_0 : \sigma_0$, we can apply Axiom C* to M_0 for the variables

$Z_0, Z_2, \ldots, Z_h$. Thus, if we let $\tau'_i \equiv [\rho'/Y]\tau_i^Y$, and we take

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

we get the desired result, as

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

and

$$\begin{array}{ll} [\tau/X]\, N \equiv [\rho'/Z_0,\ \tau/Z_1,\ \tau'_2/Z_2,\ \ldots,\ \tau'_h/Z_h]\, M_0 & \text{since } X \notin FV(\rho') \\ \qquad \equiv [\rho'/Y]\,[Y/Z_0,\ \tau/Z_1,\ \tau_2^Y/Z_2,\ \ldots,\ \tau_h^Y/Z_h]\, M_0 & \\ & \text{by rearranging substitutions} \\ \qquad \equiv [\rho'/Y]\, M' & M_0 \text{ is the generalizer of } M, M' \\ \qquad \equiv N' & \text{by (3)} \quad \blacksquare \end{array}$$

Note that the property of preserving type substitution structure does **not** hold for F-equality. Backward β_2 reduction causes problems as witnessed by the following counter-example. Assume x has type $\forall Y.Y$, and take $M \equiv xX$ with $\tau \equiv \sigma_1 \to \sigma_2$ and $N' \equiv (\lambda Z.x(Z \to \sigma_2))\sigma_1$. Then,

$$[\tau/X]M \equiv x(\sigma_1 \to \sigma_2) \quad {}_{\beta_2}\!\longleftarrow \quad (\lambda Z.x(Z \to \sigma_2))\sigma_1 \equiv N'$$

Now, since $\tau \equiv \sigma_1 \to \sigma_2$ 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 τ. 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 η_2 reduction. Proofs of thes are straightforward.

Fact 6.2
If $[\tau/X]M \longrightarrow_F N'$ *then there exists a term* N *such that* $N' \equiv [\tau/X]N$ *and* $M \longrightarrow_F N$.

Fact 6.3
If $[\tau/X]M\ {}_{\eta_2}\!\longleftarrow N'$ *then there exists a term* N *such that* $N' \equiv [\tau/X]N$ *and* $M\ {}_{\eta_2}\!\longleftarrow N$.

7 Weak Genericity

In this section, we prove two weaker forms of the Genericity Theorem, which will be used in the final result. The weakness or asymmetry arises because $=_F$ and $=_{c^*}$ are used respectively in the premise instead of $=_{Fc}$. Generalizers are a key tool in the proof with $=_F$. We first need the following lemma about simultaneous substitutions.

Lemma 7.1
Given type σ, if $[\tau_1/X_1, \ldots, \tau_n/X_n]\,\sigma = [\rho_1/X_1, \ldots, \rho_n/X_n]\,\sigma$ and $\tau_i \neq \rho_i$ for some $1 \leq i \leq n$, then X_i is not free in σ.

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

Theorem 7.2 (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 τ, 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.
Furthermore, since reduction is type-substitutive[1], 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

$$[\tau/X]M_1' \;\equiv\; [\tau/X]M_2' \tag{4}$$

We are in the case $\tau = \rho$ so case 1) of the definition of generalizer applies, i.e. $h = 1$.
Thus, choose fresh type variables Z_0, Z_1, Z_2.
Then, by Term Generalization (Theorem 5.5), there exist unique Z_0, Z_1, Z_2-generalizers: Γ_0 of Γ, Γ; M_0' of M_1', M_2'; and σ_0 of σ, σ, such that $\Gamma_0 \vdash M_0' : \sigma_0$.
By definition of generalizer, we have

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

Now, by the substitution convention applied to (4), $X \notin FV(\tau)$.
So, in particular, $\tau \neq X$. We can thus apply lemma 7.1 to the above two equations to obtain that Z_0 and Z_1 are not free in Γ_0 and σ_0.
Hence, we can apply Axiom C* to M_0' for Z_0, Z_1 in the following:

$$\begin{array}{lll} M_1 =_F & M_1' & M_1' \text{ is the normal form of } M_1 \\ \quad\equiv & [X/Z_0, \tau/Z_1, X/Z_2]\, M_0' & M_0' \text{ is the generalizer of } M_1', M_2' \\ \quad =_{Fc} & [\tau/Z_0, X/Z_1, X/Z_2]\, M_0' & \text{by Axiom C* for } Z_0, Z_1 \\ \quad\equiv & M_2' & M_0' \text{ is the generalizer of } M_1', M_2' \\ \quad =_F & M_2 & M_2' \text{ is the normal form of } M_2 \end{array}$$ ∎

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

Theorem 7.3 (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* τ, *then* $M_1 =_{Fc} M_2$.

Proof: Apply Quasi-Genericity of C*-equality (Theorem 6.1) 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 σ, the type of M_1 and M_2. Apply now Weak Genericity of F-equality (Theorem 7.2) to $[\tau/X]N \equiv [\tau/X]M_2$. Then, $N =_{Fc} M_2$. Hence, $M_1 =_{Fc} N =_{Fc} M_2$. ■

We will not use the two corollaries below but they clarify where we are in the overall proof, namely at weak forms of the Genericity Theorem.

Corollary 7.4
Let $\Gamma \vdash M, N : \forall X.\sigma$. *If* $M\tau =_F N\tau$ *for some type* τ, *then* $M =_{Fc} N$.

Corollary 7.5
Let $\Gamma \vdash M, N : \forall X.\sigma$. *If* $M\tau =_{c^*} N\tau$ *for some type* τ, *then* $M =_{Fc} N$.

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 β_1, β_2, and η_1 reductions but **not** with η_2 reduction. To see this last point, take M of type $\forall Z.\sigma$ with $Z \notin FV(\sigma)$, and X fresh. Then,

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

but $\lambda X.M\tau$ does not η_2-reduce to M.

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

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.

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

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 definition of Axiom C*, there exist ν, N_1, ρ, ρ', 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 $\Gamma \vdash \lambda x\!:\!\nu.N_1 : \nu \to \sigma$, and Y not free in Γ or $\nu \to \sigma$.
Clearly, Y is also not free in ν. Hence, $\mu = [\rho/Y]\nu = \nu$.
Moreover, Y is not free in σ, 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 $(\lambda x\!:\!\mu.M_1)M_2$.
Then, by 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 Γ or σ.
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

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

and $[\rho'/Y]((\lambda x\!:\!\nu.N_1)N_2) \xrightarrow{1}_{\beta_1} [\rho'/Y][N_2/x]N_1$, since β_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, $[\mu/X]M_1 \ =_{c^*}\ [\mu/X]N_1$.

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

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

Then, by definition of Axiom C*, there exist N_1, ρ, ρ', 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 Γ or $\forall X.\sigma$.

Clearly, Y is not free in σ, 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 definition of Axiom C*, there exist N_1, ν, ρ, ρ', 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 Γ or σ.

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

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

and $[\rho'/Y]\,((\lambda X.N_1)\nu) \ \xrightarrow{1}_{\beta_2}\ [\rho'/Y][\nu/X]N_1$, since β_2 is substitutive.

Case: $\lambda x\!:\!\mu.M_1x \ \xrightarrow{1}_{\eta_1}\ M_1$ 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 $\lambda x\!:\!\mu.M_1x \ \stackrel{1}{=}_{c^*}\ \lambda x\!:\!\mu.N_1x$.

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, $\lambda x\!:\!\mu.N_1x \ \xrightarrow{1}_{\eta_1}\ N_1$.

Therefore, take $N' \equiv N_1$.

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

Then, by definition of Axiom C*, there exist N_1, ρ, ρ', 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 σ.

Clearly, Y is also not free in $\mu \to \sigma$, 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 definition of Axiom C*, there exist ν, N_1, ρ, ρ', 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 $\mu = [\rho/Y]\nu$

$M_1 = [\rho/Y]N_1$, $\Gamma \vdash \lambda x\!:\!\nu.N_1x : \nu \to \sigma$, and Y not free in Γ or $\nu \to \sigma$.

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

Also, Y is not free in σ, 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)

If
$$\begin{tikzcd} M \arrow[d, "\beta_1\beta_2\eta_1"'] & =_{c^*} & N \\ M' & & \end{tikzcd}$$
then there exists a term N' *such that*
$$\begin{tikzcd} M \arrow[d, "\beta_1\beta_2\eta_1"'] & =_{c^*} & N \arrow[d, "\beta_1\beta_2\eta_1"] \\ M' & =_{c^*} & N' \end{tikzcd}$$

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:

$$\begin{tikzcd}
M \arrow[d, "1", "\beta_1\beta_2\eta_1"'] & \stackrel{1}{=}_{c^*} & & N_1 \arrow[d, "1"', "\beta_1\beta_2\eta_1"] & \stackrel{1}{=}_{c^*} & \ldots & \stackrel{1}{=}_{c^*} & N \arrow[d, "1"', "\beta_1\beta_2\eta_1"] \\
M_1' \arrow[d, "1", "\beta_1\beta_2\eta_1"'] & \stackrel{1}{=}_{c^*} & N_{11} \arrow[d, "1"', "\beta_1\beta_2\eta_1"] \stackrel{1}{=}_{c^*}\ldots\stackrel{1}{=}_{c^*} & N_{1i} & & \ldots & & \vdots \\
M_2' & \stackrel{1}{=}_{c^*}\ldots\stackrel{1}{=}_{c^*} & N_{2j} & \ldots & & \ldots & & \\
\vdots \arrow[d, "1", "\beta_1\beta_2\eta_1"'] & & & & & & & \vdots \arrow[d, "1"', "\beta_1\beta_2\eta_1"] \\
M' =_{c^*} \ldots & & & \ldots & & \ldots & & =_{c^*} N'
\end{tikzcd}$$

■

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 (η_2-postponement)
If $M \longrightarrow_F M'$ *then there exists a term* M'' *such that* $M \longrightarrow_{\beta_1\beta_2\eta_1} M'' \longrightarrow_{\eta_2} M'$.

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* τ*, then* $M =_{Fc} N$*.*

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

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

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.

<u>*Case:*</u> The chain consists entirely of F-equalities, i.e., $[\tau/X]M =_F [\tau/X]N$. Then, by Weak Genericity of F-equality (Theorem 7.2), we have the result $M =_{Fc} N$.

<u>*Case:*</u> The chain consists entirely of C*-equalities, i.e., $[\tau/X]M =_{c^*} [\tau/X]N$. Then, by Weak Genericity of C*-equality (Theorem 7.3), $M =_{Fc} N$.

<u>*Case:*</u> 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 η_2-postponement (Lemma 9.1) to the reduction sequence from M''_1. Thus, there exist terms M'_1, M'_3, N'_1 such that:

$$\begin{tikzcd}
{[\tau/X]M} \arrow[ddrr, "F"'] & & & & M''_1 \arrow[dl, "\beta_1\beta_2\eta_1"'] & =_{c^*} & M''_2 \arrow[dr, "F"] & & M''_3 \arrow[dl, "F"] \\
 & & & M'_1 \arrow[dl, "\eta_2"] & & & & M'_3 & \\
 & & N'_1 & & & & & &
\end{tikzcd}$$

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

$$\begin{tikzcd}
{[\tau/X]M} \arrow[ddrr, "F"'] & & & & M''_1 \arrow[dl, "\beta_1\beta_2\eta_1"'] & =_{c^*} & M''_2 \arrow[dl, "\beta_1\beta_2\eta_1"] \arrow[dr, "F"] & & M''_3 \arrow[dl, "F"] \\
 & & & M'_1 \arrow[dl, "\eta_2"] & =_{c^*} & M'_2 & & M'_3 & \\
 & & N'_1 & & & & & &
\end{tikzcd}$$

The Church-Rosser property can then be used to complete the diamond between M_2' and M_3':

$$\begin{tikzcd}
{[\tau/X]M} \arrow[ddrr, "F"'] & & & & M_1'' \arrow[dl, "\beta_1\beta_2\eta_1"'] & =_{c^*} & M_2'' \arrow[dl, "\beta_1\beta_2\eta_1"] \arrow[dr, "F"] & & M_3'' \arrow[dl, "F"] \\
& & & M_1' \arrow[dl, "\eta_2"] & =_{c^*} & M_2' \arrow[dr, "F"'] & & M_3' \arrow[dl, "F"] & \\
& & N_1' & & & & N_3' & &
\end{tikzcd}$$

In this way, the original path of Fc-equalities from $[\tau/X]M$ to M_3'' can be replaced by;

$$\begin{tikzcd}
{[\tau/X]M} \arrow[dr, "F"'] & & M_1' \arrow[dl, "\eta_2"] & =_{c^*} & M_2' \arrow[dr, "F"'] & & M_3'' \arrow[dl, "F"] \\
& N_1' & & & & N_3' &
\end{tikzcd}$$

Now 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:

$$\begin{tikzcd}
{[\tau/X]M} \arrow[dr, "F"'] & & M_1' \arrow[dl, "\eta_2"] & =_{c^*} & M_2' \arrow[dr, "F"'] & & \cdots & & M_{n-1}' \arrow[dl, "\eta_2"] & =_{c^*} & M_n' \arrow[dr, "F"'] & & {[\tau/X]N} \arrow[dl, "F"] \\
& N_1' & & & & {} & & {} & & & & N' &
\end{tikzcd}$$

where each left-pointing arrow, except for the final one, consists of forward η_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.

Consider now the start of the transformed chain:

$$\begin{tikzcd}
{[\tau/X]M} \arrow[dr, "F"'] & & M_1' \arrow[dl, "\eta_2"] & =_{c^*} & M_2' \\
& N_1' & & &
\end{tikzcd}$$

— By Fact 6.2, there exists N_1 such that $N_1' \equiv [\tau/X]N_1$ and $M \longrightarrow_F N_1$.

— By Fact 6.3, there exists M_1 such that $M_1' \equiv [\tau/X]M_1$ and $N_1 \;{}_{\eta_2}\!\!\longleftarrow M_1$.

— By Quasi-Genericity of C*-equality (Theorem 6.1), there exists M_2 such that $M_2' \equiv [\tau/X]M_2$ and $M_1 =_{c^*} M_2$.

Thus, we have, for the first three links of the transformed chain,

$$
\begin{tikzcd}
{[\tau/X]M} \arrow[dr, "F"'] & & {[\tau/X]M_1 \equiv M'_1} \arrow[dl, "\eta_2"] & =_{c^*} & {M'_2 \equiv [\tau/X]M_2} \\
& {N'_1 \equiv [\tau/X]N_1} & & &
\end{tikzcd}
$$

with $M \longrightarrow_F N_1 \;{}_{\eta_2}\!\!\longleftarrow M_1 =_{c^*} M_2$, i.e., $M =_{Fc} M_2$.

Iterate then this process along the transformed chain from $M'_2 \equiv [\tau/X]M_2$, and thus "push" the type substitution $[\tau/X]$ along the chain. Finally, we end up with a term M_n such that $M =_{Fc} M_n$ and $[\tau/X]M_n \equiv M'_n$.

Apply now Weak Genericity of F-equality (Theorem 7.2) to $[\tau/X]M_n \equiv M'_n =_F [\tau/X]N$. 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* τ, *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, by the Main Lemma (Lemma 9.2), $MZ =_{Fc} NZ$.
Observe that Z fresh implies 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.
Hence, by ξ_2, $\lambda Z.MZ =_{Fc} \lambda Z.NZ$, and, by η_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 on an arbitrary (partial) combinatory algebra $(D, .)$, i.e., on a model of (partial) Combinatory Logic. That is, 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, $(D, .)$ yields a model of the type free λ-calculus $(D, ., [\![-]\!])$, 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 $[\![er(M)]\!]_\xi$, i.e., the interpretation in D, under term environment ξ, of the type-free term $er(M)$. A result in [Mit86] (see also [CL91]) shows that the meaning of an arbitrary term M of system F in the PER model, is given by the equivalence class of the meaning of its erasure, in the p.e.r. which interprets its type. More formally, if environment ξ' is obtained from ξ by forgetting type information,

$$[\![\Gamma \vdash M : \sigma]\!]_\xi = \{[\![er(M)]\!]_{\xi'}\}_{[\![\sigma]\!]}$$

It is then clear that PER models realize Axiom C: if $M\tau$ and $M\tau'$ live in the same type σ, then their meanings are identical as $er(M\tau) = er(M\tau')$. Note that in these few lines, we have omitted to mention the interpretation of type variables which may occur in types; indeed, and more generally, types are maps from (the objects of) PER to (the objects of) PER.

As for dI-based models, we only recall here 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].[2] (The impossibility of viewing types as functors, in general, was discussed in the introduction, in view of the the (contra-) and (co-)variance of the $\rightarrow$ functor.) In short, let $F : \mathrm{DI}^L \rightarrow \mathrm{DI}^L$ be a functor. Then ΠF, 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 σ. This means that σ 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, ΠF 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 (AD) below. Following [AL91], let $c = (c_0, c_1)$ be an internal category to a Cartesian Closed Category E with all finite limits. Let c^{c_0} be

[2] A rigid embedding is a projection with injection j and surjection j^R, which is the identity below the image of j.

the category of internal functors. Then (E, c) yields a model of system F if there exists the (internal) product functor $\Pi : c^{c_0} \to c$ as the right adjoint of the (internal) diagonal functor $K : c \to c^{c_0}$, i.e., the functor that to each A associates the functor KA, which is constant A. In other words

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

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

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

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

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

This is equivalent, in these models, to the isomorphism $\Pi(KA) \cong A$, 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 hard part now comes with the semantics of the Genericity Theorem. Indeed,

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

is not an equation, but an implication between equations. Thus a model $\mathcal{M}$ of Fc does not need to realize it, in either of the two senses

$$\begin{array}{l} (1) \quad \mathcal{M} \models \exists\tau\ M\tau = N\tau \quad \Rightarrow \quad \mathcal{M} \models M = N \\ \text{or} \\ (2) \quad \mathcal{M} \models (\exists\tau\ M\tau = N\tau \ \Rightarrow\ M = N) \end{array}$$

For example, over PER models or dI-domains, consider $0, K : \forall X.X \to (X \to X)$. Take then a type τ which has at most one element, for example, $\forall X.X$ or $\forall X.\forall Y.X \to (Y \to X)$. Then in both classes of models $K\tau = 0\tau$, but, of course, $K \neq 0$. We have not yet found models of the "genericity" implication (GEN) in either form, in spite of the many models of Fc and the provability of the implication. Note that their semantic understanding 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 are based on extending core calculi by, possibly, more equation schemes. Since (GEN) is not preserved in equational varieties, the investigation of which equational theories realize it is a further theoretical challenge posed by the Genericity Theorem.

Acknowledgments

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 for valuable suggestions about this paper.

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

References

[ACC92] M. Abadi, L. Cardelli, P.-L. Curien. "Formal Parametric Polymorphism." Proceedings of the 20th Symposium on *Principles of Programming Languages*, Charleston, January 1993.

[AL91] A. Asperti, 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, P.J. Scott. "Functorial Polymorphism." *Theoretical Computer Science* 70, pages 35–64, January 1990. Corrigendum in 71, page 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, R. Statman. *Typed Lambda Calculus with Applications.* In preparation.

[CL91] L. Cardelli, G. Longo. "A Semantic Basis for Quest." *Journal of Functional Programming* 1, pages 417–458, April 1991.

[CMMS91] L. Cardelli, S. Martini, J.C. Mitchell, A. Scedrov. "An Extension of System F with Subtyping." 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, G. Longo. "A Calculus for Overloaded Functions with Subtyping." Proceedings of the Conference on *LISP and Functional Programming*, San Francisco, July 1992. Extended abstract.

[CGW88] T. Coquand, C.A. Gunter, G. Winskel. "DI-Domains as a Model of Polymorphism." Proceedings of the 3rd Workshop on *Mathematical Foundations of Programming Language Semantics*, New Orleans, April 1987. Lecture Notes in Computer Science 298, pages 344–363. Springer-Verlag, 1988. Edited by M. Main, A. Melton, M. Mislove, and D. Schmidt.

[DL89] R. Di Cosmo, G. Longo. "Constructively Equivalent Propositions and Isomorphisms of Objects (or Terms as Natural Transformations)." Proceedings of the Workshop on *Logic for Computer Science*, Berkeley, November 1989. Mathematical Sciences Research Institute Publications 21. Springer-Verlag, 1992. Edited by Y.N. Moschovakis.

[EK66] S. Eilenberg, G.M. Kelly. "A Generalization of the Functorial Calculus." *Journal of Algebra* 3, pages 366–375, 1966.

[FGSS88] P.J. Freyd, J.-Y. Girard, A. Scedrov, P.J. Scott. "Semantic Parametricity in Polymorphic Lambda-Calculus." Proceedings of the 3rd Symposium on *Logic in Computer Science*, Edinburgh, Scotland, June 1988.

[Gir71] J.-Y. Girard. "Une Extension de l'Interpretation Fonctionelle de Gödel à l'Analyse et son Application à l'Elimination des Coupures dans l'Analyse et la Théorie des Types." Proceedings of the 2nd *Scandinavian Logic Symposium*, pages 63–92. 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, pages 159–192, 1986.

[GLT89] J.-Y. Girard, Y. Lafont, P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1989.

[GSS] J.-Y. Girard, A. Scedrov, P.J. Scott. "Normal Forms and Cut-Free Proofs as Natural Transformations." Proceedings of the Workshop on *Logic for Computer Science*, Berkeley, November 1989. Mathematical Sciences Research Institute Publications 21, pages 217–241. Springer-Verlag, 1992. Edited by Y.N. Moschovakis.

[Hyl] M. Hyland. "A Small Complete Category." *Annals of Pure and Applied Logic* 40, pages 135–165, 1988.

[LM91] G. Longo, E. Moggi. "Constructive Natural Deduction and its ω-set Interpretation." *Mathematical Structures in Computer Science* 1 (2), pages 215–253, 1991.

[LS86] J. Lambek, P.J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics 7. Cambridge University Press, 1986.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[MR91] Q. Ma, J.C. Reynolds. "Types, Abstraction, and Parametric Polymorphism: Part 2." Proceedings of the Workshop on *Mathematical Foundations of Programming Language Semantics*, March 1991. Lecture Notes in Computer Science. Springer-Verlag, 1991. Edited by S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt.

[Mit86] J.C. Mitchell. "A Type Inference Approach to Reduction Properties and Semantics of Polymorphic Expressions." Proceedings of the Conference on *LISP and Functional Programming*, 1986.

[Rey74] J.C. Reynolds. "Towards a Theory of Type Structure." Proceedings of *Colloque sur la Programmation.* Lecture Notes in Computer Science 19, pages 408–425. Springer-Verlag, 1974. Edited by B. Robinet.

[Rey83] J.C. Reynolds. "Types, Abstraction and Parametric Polymorphism." *Information Processing* 83, pages 513–523. North-Holland, 1983. Edited by R.E.A. Mason.

[Sco72] D. Scott. "Continuous lattices." *Toposes, Algebraic Geometry and Logic*, Lecture Notes in Mathematics 274, pages 97–136. Springer-Verlag, 1972. Edited by F.W. Lawvere.

[SP82] M. Smyth, G. Plotkin. "The Category Theoretic Solution of Recursive Domain Equations." *SIAM Journal of Computing* 11, pages 761–783, 1982.

[Str67] C. Strachey. "Fundamental concepts in programming languages." Unpublished lecture notes from the International Summer School in Computer Programming, Copenhagen, August 1967.