Reynolds's Parametricity Theorem*[†]

Robert Harper

Spring, 2024

1 Introduction

In a landmark paper Reynolds (1983) develops a mathematical account of Strachey's informal concept of *parametricity* of polymorphic functions. Parametricity characterizes the "uniform" behavior of polymorphic functions using *logical relations*, a concept introduced by Tait (1967) in the study of functionals of finite type. Reynolds's work, which was motivated by the study of data abstraction in programming languages, was done around the same time as, and independently of, Girard's extension of Tait's method to second-order quantification, which was motivated by the analysis of proofs in second-order logic.[1] Whereas Girard made use of unary predicates, Reynolds used binary relations, a technically small, yet practically large, difference that gave rise to new methods for proving properties of programs knowing only their types. Reynolds observed that the type discipline of a language determines the abstraction properties enjoyed by its programs; in particular, clients of abstract types are *polymorphic*, and hence enjoy stability properties across changes of representation determined entirely by their types.

The formulation given here makes use of the aforementioned ideas from Tait, Girard, and Reynolds, but cast in an operational framework. In contrast to the presentation in PFPL the formulation given here is independent of a prior notion of equality. In compensation candidates are required to enjoy a property called *zig-zag completeness* (Krishnaswami and Dreyer, 2013), which suffices to ensure that equality is symmetric and transitive by a simple and direct argument.

2 The Language

Please see Harper (2020) for the definition of the language F, including its syntax and dynamics. The dynamics is to be understood via a tacit *erasure* of type information from terms given as follows:

1. Type labels are removed from λ-abstractions, $\lambda(x.M)$ becomes $\lambda(x.M)$.
2. Λ-abstractions $\Lambda(X.M)$ are replaced by anonymous Λ-abstractions $\Lambda(M)$.
3. Type applications $\mathsf{Ap}(M;A)$ are replaced by instantiations $\mathsf{Ap}(M)$ corresponding to $\Lambda(N)$.

With this understanding it is never necessary to substitute types for type variables when defining the dynamics of terms.

*Copyright © Robert Harper. All Rights Reserved

†Thanks to Carlo Angiuli, Karl Crary, Yiyang Guo, and Harrison Grodin for helpful discussions.

[1]See Harper (2020) for an introduction to Girard's method.

$$\begin{tikzcd} M \arrow[r, no head] \arrow[dr, dashed, no head] & M' \\ N \arrow[r, no head] \arrow[ur, no head] & N' \end{tikzcd}$$

Figure 1: Zig-Zag Completeness

3 Parametricity

A binary relation $R : A \leftrightarrow A'$ between closed terms of closed types A and A' is a *parametricity candidate for A and A'*, written $R : A \leftrightarrow A'$, iff it satisfies the following two closure conditions:

1. *Head expansion*: if $M\ R\ M'$, then
   (a) if $N \longmapsto M$, then $N\ R\ M'$, and
   (b) if $N' \longmapsto M'$, then $M\ R\ N'$.
2. *Zig-Zag Completeness (ZZC)*: if $M\ R\ M'$, $N\ R\ N'$, and $N\ R\ M'$, then $M\ R\ N'$.

Head expansion is a natural requirement when thinking of types as specifications of program behavior. Zig-zag completeness is depicted in Figure 1. It may be stated in terms of the converse and composition of relations as the containment $R \circ R^{\mathsf{op}} \circ R \subseteq R$. The opposite containment is always valid, so zig-zag completeness may be re-stated as the equation $R \circ R^{\mathsf{op}} \circ R = R$.

A *candidate assignment*, η, for type substitutions $\delta, \delta' : \Delta$ is a function assigning to each $\Delta \vdash X$ type a candidate $\eta(X) : \delta X \leftrightarrow \delta' X$.

Definition 1 (Similarity)**.** *The binary relation, $M \sim M' \in A\ [\eta : \delta \leftrightarrow \delta']$, called* similarity*, is defined for substitutions $\delta, \delta' : \Delta$, closed terms $M : \hat{\delta}(A)$ and $M' : \widehat{\delta'}(A)$, and candidate assignment $\eta : \delta \leftrightarrow \delta'$, by induction on the structure of A as follows:*

$$M \sim M' \in \mathsf{ans}\ [\eta : \delta \leftrightarrow \delta'] \text{ iff } M, M' \longmapsto^* \mathsf{yes} \text{ or } M, M' \longmapsto^* \mathsf{no}$$

$$M \sim M' \in X\ [\eta : \delta \leftrightarrow \delta'] \text{ iff } M\ \eta(X)\ M'$$

$$M \sim M' \in A_1 \to A_2\ [\eta : \delta \leftrightarrow \delta'] \text{ iff } \begin{cases} M \longmapsto^* \lambda(x.M_2),\ M' \longmapsto^* \lambda(x.M_2'), \\ M_1 \sim M_1' \in A_1\ [\eta : \delta \leftrightarrow \delta'] \text{ implies} \\ [M_1/x]M_2 \sim [M_1'/x]M_2' \in A_2\ [\eta : \delta \leftrightarrow \delta'] \end{cases}$$

$$M \sim M' \in \forall(X.A_2)\ [\eta : \delta \leftrightarrow \delta'] \text{ iff } \begin{cases} M \longmapsto^* \Lambda(M_2),\ M' \longmapsto^* \Lambda(M_2'), \text{and if } R : B \leftrightarrow B' \text{ then} \\ M_2 \sim M_2' \in A_2\ [\eta[X \mapsto R] : \delta[X \mapsto B] \leftrightarrow \delta'[X \mapsto B']] \end{cases}$$

It is immediate from the form of the definition that similarity is closed under head expansion, given that candidates are required to be.

Exercise 1. *Extend the definition of similarity to account for existential types, $\exists(X.A)$, defined as in Harper (2016). Your definition should account for the possibility that related packages need not have the same implementation type, rather the operations need only correspond.*

Lemma 2 (Head Expansion)**.** *If $M \sim M' \in A\ [\eta : \delta \leftrightarrow \delta']$, then if $N \longmapsto M$, then $N \sim M' \in A\ [\eta : \delta \leftrightarrow \delta']$, and if $N' \longmapsto M'$, then $M \sim N' \in A\ [\eta : \delta \leftrightarrow \delta']$.*

Proof. All conditions are defined in terms of evaluation to a value, and candidates are assumed to be closed under head expansion. □

Lemma 3 (Zig-Zag Completeness)**.** *For each $\Delta \vdash A$ type, and each candidate assignment η for $\delta, \delta' : \Delta$, the similarity relation $\_ \sim \_ \in A\ [\eta : \delta \leftrightarrow \delta']$ is zig-zag complete.*

Proof. By induction on the structure of A:

1. If A is a type variable X, then $\eta(X)$ is a candidate for $\delta(X)$ and $\delta'(X)$, and is therefore zig-zag complete.

2. If $A = \forall(X.A_2)$. Assume that

   (a) $M \sim M' \in A\ [\eta : \delta \leftrightarrow \delta']$,
   (b) $N \sim N' \in A\ [\eta : \delta \leftrightarrow \delta']$, and
   (c) $N \sim M' \in A\ [\eta : \delta \leftrightarrow \delta']$.

   It follows that

   (a) $M \longmapsto^* \Lambda(M_2)$,

   (b) $M' \longmapsto^* \Lambda(M'_2)$,

   (c) $N \longmapsto^* \Lambda(N_2)$, and

   (d) $N' \longmapsto^* \Lambda(N'_2)$.

   To show that $M \sim N' \in A\ [\eta : \delta \leftrightarrow \delta']$, it suffices to assume that $R : A \leftrightarrow A'$, and show that $M_2 \sim N'_2 \in A_2\ [\eta[X \longmapsto R] : \delta[X \longmapsto B] \leftrightarrow \delta'[X \longmapsto B']]$.

   From the assumptions we have

   (a) $M_2 \sim M'_2 \in A_2\ [\eta[X \longmapsto R] : \delta[X \longmapsto A] \leftrightarrow \delta'[X \longmapsto A']]$,
   (b) $N_2 \sim N'_2 \in A_2\ [\eta[X \longmapsto R] : \delta[X \longmapsto A] \leftrightarrow \delta'[X \longmapsto A']]$, and
   (c) $N_2 \sim M'_2 \in A_2\ [\eta[X \longmapsto R] : \delta[X \longmapsto A] \leftrightarrow \delta'[X \longmapsto A']]$.

   But then the result follows by the inductive hypothesis that similarity at A_2 relative to $\eta[X \longmapsto R]$ is zig-zag complete.

□

Exact equality of two terms in a type, $\Gamma \gg_\Delta M \doteq M' \in A$, is defined to mean that for all substitutions $\delta, \delta' : \Delta$, for all candidate assignments η for δ and δ', if $\gamma \sim \gamma' \in \Gamma\ [\eta : \delta \leftrightarrow \delta']$, then $\widehat{\gamma}(M) \sim \widehat{\gamma'}(M) \in A\ [\eta : \delta \leftrightarrow \delta']$. Write $\Gamma \gg_\Delta M \in A$ to mean $\Gamma \gg_\Delta M \doteq M \in A$.

Lemma 4 (Compositionality). *Suppose that $\Delta \vdash B$ type and $\Delta, X \vdash A$ type. Let $\delta, \delta' : \Delta$ and assume $\eta : \delta \leftrightarrow \delta'$. Then,*

$$M \sim M' \in [B/X]A\ [\eta : \delta \leftrightarrow \delta'] \text{ iff } M \sim M' \in A\ [\eta_1 : \delta_1 \leftrightarrow \delta_1'],$$

where $\delta_1 = \delta[X \longmapsto \hat{\delta}B]$, and $\delta_1' = \delta'[X \longmapsto \widehat{\delta'}B]$, and $\eta_1 = \eta[X \longmapsto \_ \sim \_ \in B\ [\eta : \delta \leftrightarrow \delta']]$.

Exercise 2. *Prove Lemma 4 by induction on the structure of A.*

In the context of F the reflexivity theorem is known as the parametricity theorem.

Theorem 5 (Parametricity). *If $\Gamma \vdash_\Delta M : A$, then $\Gamma \gg_\Delta M \in A$.*

Proof. By induction on typing derivations, using Lemma 2 and 4. Here are two cases of the proof.

1. Consider $\Gamma \vdash_\Delta \Lambda(M) : \forall(X.A)$ following from $\Gamma \vdash_{\Delta, X \,\mathsf{type}} M : A$ by the introduction for the quantifier. Suppose that $\delta, \delta' : \Delta$ are instances of Δ, that $\eta : \delta \leftrightarrow \delta'$, and that $\gamma \sim \gamma' \in \Gamma\ [\eta : \delta \leftrightarrow \delta']$. The goal is to show that $\Lambda(\hat{\gamma}(M)) \sim \Lambda(\widehat{\gamma'}(M')) \in \forall(X.A)\ [\eta : \delta \leftrightarrow \delta']$. To this end suppose that B type, B' type, and $R : B \leftrightarrow B'$; it suffices to show that $\hat{\gamma}(M) \sim \widehat{\gamma'}(M') \in A\ [\eta[X \longmapsto R] : \delta[X \longmapsto B] \leftrightarrow \delta'[X \longmapsto B']]$. But this follows directly by induction, noting that $\eta[X \longmapsto R] : \delta[X \longmapsto B] \leftrightarrow \delta'[X \longmapsto B']$ is again a candidate assignment.

2. Consider $\Gamma \vdash_\Delta \mathsf{Ap}(M) : [B/X]A$ following from $\Gamma \vdash_\Delta M : \forall(X.A)$ and $\Delta \vdash B$ type by the elimination rule for the quantifier. Suppose that $\delta, \delta' : \Delta$ and $\eta : \delta \leftrightarrow \delta'$, and that $\gamma \sim \gamma' \in \Gamma\ [\eta : \delta \leftrightarrow \delta']$, with the goal to show that

   $$\mathsf{Ap}(\hat{\gamma}(M)) \sim \mathsf{Ap}(\widehat{\gamma'}(M')) \in [B/X]A\ [\eta : \delta \leftrightarrow \delta'].$$

   Note that by the second assumption $\hat{\delta}(B)$ type and $\widehat{\delta'}(B)$ type, written $\widehat{B}$ and $\widehat{B'}$, respectively. By Lemma 4 the objective becomes to show that

   $$\mathsf{Ap}(\hat{\gamma}(M)) \sim \mathsf{Ap}(\widehat{\gamma'}(M')) \in A\ [\eta[X \longmapsto R] : \delta[X \longmapsto \widehat{B}] \leftrightarrow \delta'[X \longmapsto \widehat{B'}]],$$

   where $R(N, N') \overset{\text{def}}{=} N \sim N' \in B\ [\eta : \delta \leftrightarrow \delta']$ for $N : \widehat{B}$ and $N' : \widehat{B'}$. By the inductive hypothesis and definition of similarity at quantified types,

   (a) $\hat{\gamma}(M) \Downarrow \Lambda(N)$;
   (b) $\widehat{\gamma'}(M) \Downarrow \Lambda(N')$;
   (c) $N \sim N' \in A\ [\eta[X \longmapsto R] : \delta[X \longmapsto \widehat{B}] \leftrightarrow \delta'[X \longmapsto \widehat{B'}]]$,

   from which the result follows by head expansion.

□

Exercise 3. *Prove the remaining cases of the parametricity theorem.*

4 Equational Soundness

Judgmental equality for F is the extension of that for simple types (see Harper (2022)) with the following two rules for quantified types:

$$\forall{-}\beta \quad \frac{\Gamma \vdash_{\Delta, X\,\mathsf{type}} M : A \qquad \Delta \vdash B\,\mathsf{type}}{\Gamma \vdash_{\Delta} \mathsf{Ap}(\Lambda(X.M); B) \equiv [B/X]M : [B/X]A} \qquad \forall{-}\eta \quad \frac{\Gamma \vdash_{\Delta} M : \forall(X.A)}{\Gamma \vdash_{\Delta} M \equiv \Lambda(X.\,\mathsf{Ap}(M;X)) : \forall(X.A)}$$

In addition the evident compatibility rules are included as well.

Theorem 6 (Soundness). *If* $\Gamma \vdash_{\Delta} M \equiv M' : A$*, then* $\Gamma \gg_{\Delta} M \doteq M' \in A$

Sketch. Closure under head expansion suffices for the β principles:

1. If $\Gamma, x : A_1 \gg_{\Delta} M_2 \in A_2$ and $\Gamma \gg_{\Delta} M_1 \in A_1$, then $\Gamma \gg_{\Delta} \lambda(x.M_2)(M_1) \doteq [M_1/x]M_2 \in A_2$.
2. If $\Gamma \gg_{\Delta, X\,\mathsf{type}} M_2 \in A_2$ and $\Delta \vdash A_1$ type, then $\Gamma \gg_{\Delta} \mathsf{Ap}(\Lambda(M_2)) \doteq M_2 \in [A_1/X]A_2$.

The definition of similarity ensures that the η principles are valid:

1. If $\Gamma \gg_{\Delta} M \in A_1 \to A_2$, then $\Gamma \gg_{\Delta} M \doteq \lambda(x.M(x)) \in A_1 \to A_2$.
2. If $\Gamma \gg_{\Delta} M \in \forall(X.A_2)$, then $\Gamma \gg_{\Delta} M \doteq \Lambda(\mathsf{Ap}(M)) \in \forall(X.A_2)$.

Compatibility of exact equality with both forms of abstraction and application are left as exercises. Parametricity states that equality is reflexive. Symmetry and transitivity are proved as in Harper (2022), using Lemma 3 and Theorem 5, as illustrated in Figure 2. □

Exercise 4. *Flesh out the proof of Theorem 6 for the cases of symmetry and transitivity, and the compatibility rule for type abstraction.*

Just as with, say, the natural numbers in the simply typed case, the judgmental equality rules are entirely inadequate for proving any interesting exact equivalences, in particular those involving a parametricity argument.

References

Robert Harper. *Practical Foundations for Programming Languages*. Cambridge University Press, Cambridge, England, Second edition, 2016.

Robert Harper. How to (re)invent Girard's method. Unpublished lecture note, Spring 2020. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/girard.pdf`.

Robert Harper. Semantic equality for typed λ-calculus. Unpublished lecture note., July 2022. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/tlc-semeq.pdf`.

Robert Harper. How to (re)invent Tait's method. Unpublished lecture note, Spring 2024. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/tait.pdf`.

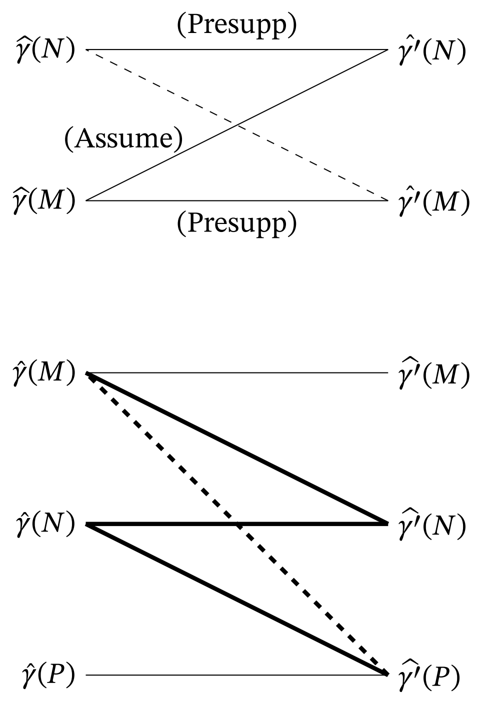

Figure 2: Symmetry and Transitivity via Zig-Zag Completeness

Neelakantan R. Krishnaswami and Derek Dreyer. Internalizing Relational Parametricity in the Extensional Calculus of Constructions. pages 20 pages, 591837 bytes, 2013. ISSN 1868-8969. doi: 10.4230/LIPICS.CSL.2013.432. URL `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2013.432`. Artwork Size: 20 pages, 591837 bytes ISBN: 9783939897606 Medium: application/pdf Publisher: Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

John C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*, pages 513–523. North-Holland/IFIP, 1983.

W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32 (2):198–212, August 1967. ISSN 0022-4812, 1943-5886. doi: 10.2307/2271658. URL `https://www.cambridge.org/core/product/identifier/S0022481200113866/type/journal_article`.