

Reynolds’s Parametricity Theorem, Directly*†

Robert Harper

Spring, 2022

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.¹ 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.² 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, 2012), which ensures that equality is transitive.

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, $\lambda_A(x . M)$ becomes $\lambda x.M$.
2. Λ -abstractions $\Lambda X.M$ are replaced by anonymous λ -abstractions $\lambda_ .M$, abbreviated $\Lambda.M$.

*Copyright © Robert Harper. All Rights Reserved

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

¹See Harper (2022b) for an introduction to Tait’s method.

²See Harper (2020) for an introduction to Girard’s method.

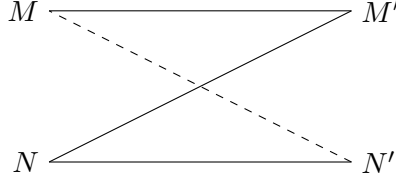


Figure 1: Zig-Zag Completeness

3. Type applications $\mathbf{Ap}(M, A)$ are replaced by ordinary applications $\mathbf{ap}(M, M_0)$ to a fixed trivial argument M_0 , abbreviated $\mathbf{Ap}(M)$.

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

3 Exact Equality Via Parametricity

A binary relation R over closed terms is a *parametricity candidate*, written $R \text{ Cand}$, iff it satisfies the following two closure conditions:

1. *Head expansion*: if $M R M'$, then
 - (a) if $N \mapsto M$, then $N R M'$, and
 - (b) if $N' \mapsto M'$, 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 $R \circ R^{\text{op}} \circ R \subseteq R$. The opposite containment is always valid, so zig-zag completeness may be re-stated as the equation $R \circ R^{\text{op}} \circ R = R$.

A *candidate assignment*, η , for Δ is a function such that $\eta(X)$ is a type candidate for each type variable X such that $\Delta \vdash X \text{ type}$.

Definition 1 (Logical Similarity). *The binary relation of logical similarity of two closed terms relative to an open type A such that $\Delta \vdash A \text{ type}$ and a candidate assignment for Δ , written $M \sim M' \in A [\eta]$, is defined by induction on the structure of A as follows:*

$$\begin{aligned}
 M \sim M' \in \text{ans} [\eta] &\Leftrightarrow \text{either } M, M' \mapsto^* \text{yes or } M, M' \mapsto^* \text{no} \\
 M \sim M' \in X [\eta] &\Leftrightarrow M \eta(X) M' \\
 M \sim M' \in A_1 \rightarrow A_2 [\eta] &\Leftrightarrow \begin{cases} M \mapsto^* \lambda x.M_2, M' \mapsto^* \lambda x.M'_2, & \text{and} \\ [M_1/x]M_2 \sim [M'_1/x]M'_2 \in A_2 [\eta] & \text{if } M_1 \sim M'_1 \in A_1 [\eta] \end{cases} \\
 M \sim M' \in \forall X.A_2 [\eta] &\Leftrightarrow \begin{cases} M \mapsto^* \Lambda.M_2, M' \mapsto^* \Lambda.M'_2, & \text{and} \\ M_2 \sim M'_2 \in A_2 [\eta[X \mapsto R]] & \text{if } R \text{ Cand} \end{cases}
 \end{aligned}$$

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

Exercise 1. *Extend the definition of logical 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 $M \sim M' \in A[\eta]$, then if $N \mapsto M$, then $N \sim M' \in A[\eta]$, and if $N' \mapsto M'$, then $M \sim N' \in A[\eta]$.*

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

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

Proof. By induction on the structure of A :

1. If A is a type variable X , note that $\eta(X)$ is required to be a parametricity candidate, and hence is zig-zag complete.
2. If $A = A_1 \rightarrow A_2$. Assume that
 - (a) $M \sim M' \in A[\eta]$,
 - (b) $N \sim N' \in A[\eta]$, and
 - (c) $N \sim M' \in A[\eta]$.

It follows that

- (a) $M \mapsto^* \lambda x.M_2$,
- (b) $M' \mapsto^* \lambda x.M'_2$,
- (c) $N \mapsto^* \lambda x.N_2$, and
- (d) $N' \mapsto^* \lambda x.N'_2$.

To show that $M \sim N' \in A[\eta]$, it suffices to assume that $M_1 \sim M'_1 \in A_1[\eta]$, and show that $[M_1/x]M_2 \sim [M'_1/x]N'_2 \in A_2[\eta]$.

From the assumptions we have

- (a) $[M_1/x]M_2 \sim [M'_1/x]M'_2 \in A_2[\eta]$,
- (b) $[M_1/x]N_2 \sim [M'_1/x]N'_2 \in A_2[\eta]$,
- (c) $[M_1/x]N_2 \sim [M'_1/x]M'_2 \in A_2[\eta]$.

But then the result follows by the induction hypothesis that similarity at A_2 relative to η is zig-zag complete.

3. If $A = \forall X.A_2$. Assume that

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

It follows that

- (a) $M \mapsto^* \Lambda.M_2$,
- (b) $M' \mapsto^* \Lambda.M'_2$,
- (c) $N \mapsto^* \Lambda.N_2$, and
- (d) $N' \mapsto^* \Lambda.N'_2$.

To show that $M \sim N' \in A [\eta]$, it suffices to assume that R is a type candidate, and show that $M_2 \sim N'_2 \in A_2 [\eta[X \mapsto R]]$.

From the assumptions we have

- (a) $M_2 \sim M'_2 \in A_2 [\eta[X \mapsto R]]$,
- (b) $N_2 \sim N'_2 \in A_2 [\eta[X \mapsto R]]$, and
- (c) $N_2 \sim M'_2 \in A_2 [\eta[X \mapsto R]]$.

But then the result follows by the inductive hypothesis that similarity at A_2 relative to $\eta[X \mapsto R]$ 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 candidate assignments η for Δ , if $\gamma \sim \gamma' \in \Gamma [\eta]$, then $\widehat{\gamma}(M) \sim \widehat{\gamma'}(M') \in A [\eta]$. *Semantic membership*, written $\Gamma \gg_{\Delta} M \in A$, is defined 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 η be a relation assignment for Δ . Then, $M \sim M' \in [B/X]A [\eta]$ iff $M \sim M' \in A [\eta[X \mapsto _ \sim _ \in B [\eta]]]$.*

Exercise 2. *Prove Lemma 4.*

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. □

Exercise 3. *Prove the parametricity theorem.*

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

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

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

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 (2022a), using Lemma 3 and Theorem 5. \square

Exercise 4. Complete the proof of Theorem 6.

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 2022a. 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 2022b. URL <https://www.cs.cmu.edu/~rwh/courses/atpl/pdfs/tait.pdf>.
- Neelakantan R Krishnaswami and Derek Dreyer. A relationally parametric model of the calculus of constructions. *Auxilliary materials at http://www.mpi-sws.org/~neelk/paradep-techreport.pdf*, 2012.
- 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. Tait. Intentional interpretations of functionals of finite type i. *Journal of Symbolic Logic*, 32: 198–212, 1967.