PFPL Supplement: Reynolds's Parametricity Theorem, Directly*

Robert Harper

Spring, 2020

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 inspired by Martin-Löf (1982) and Constable et al. (1986) and used throughout Harper (2016). 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.

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

2 The Language

Please see Harper (2020a) 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, $\lambda x{:}A.M$ becomes $\lambda x.M$.

2. Λ-abstractions $\Lambda X.M$ are replaced by anonymous λ-abstractions $\lambda\_.M$, abbreviated $\Lambda.M$.

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

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

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

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 *type candidate*, written $R\ \mathsf{Cand}$, 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 $N' \longmapsto N$, then $M\ R\ N'$.
2. *Zig-Zag Completeness (ZZC)*: if $M\ R\ M'$, $N\ R\ N'$, and $N\ R\ M'$, then $M\ R\ N'$.

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

Head expansion is a natural requirement when thinking of types as descriptive of program behavior. It is used in the proof of the Fundamental Theorem. Zig-zag completeness is depicted in Figure 1. It may be re-stated using the converse and composition of relations as the containment $R \circ R^{\mathsf{op}} \circ R \subseteq R$. The opposite containment is easily derived by going back and forth across the relation, strengthening it to an equality.

For any binary relation R, if $M\ R\ M'$, then $M\ R^{\mathsf{op}} \circ R\ M$. When R is symmmetric and transitive, $R^{\mathsf{op}} \circ R = R$, and hence R is reflexive on its field. Zig-zag completeness ensures that a similar property holds that is sufficient for our purposes.

Lemma 3.1 (Zig-Zag Lemma)**.** *If R is zig-zag complete, then so is $R^{op} \circ R$.*

Proof.

$$\begin{aligned}(R^{\mathsf{op}} \circ R) \circ (R^{\mathsf{op}} \circ R)^{\mathsf{op}} \circ (R^{\mathsf{op}} \circ R) &= (R^{\mathsf{op}} \circ R) \circ (R^{\mathsf{op}} \circ R) \circ (R^{\mathsf{op}} \circ R) \\ &= (R^{\mathsf{op}} \circ R) \circ R^{\mathsf{op}} \circ (R \circ R^{\mathsf{op}} \circ R) \\ &= R^{\mathsf{op}} \circ (R \circ R^{\mathsf{op}} \circ R) \\ &= R^{\mathsf{op}} \circ R.\end{aligned}$$

□

Similarity of two closed terms relative to an open type A such that $\Delta \vdash A\ \mathsf{type}$ and a candidate

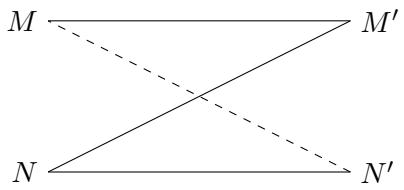

Figure 1: Zig-Zag Completeness

assignment for Δ, written $M \sim M' \in A\,[\,\eta\,]$, is defined by induction on the structure of A as follows:

$$M \sim M' \in \mathbf{2}\,[\,\eta\,] \Leftrightarrow \textit{either } M, M' \longmapsto^* \mathsf{Y} \textit{ or } M, M' \longmapsto^* \mathsf{N}$$

$$M \sim M' \in X\,[\,\eta\,] \Leftrightarrow M\ \eta(X)\ M'$$

$$M \sim M' \in A_1 \to A_2\,[\,\eta\,] \Leftrightarrow \begin{cases} M \longmapsto^* \lambda x.M_2,\ M' \longmapsto^* \lambda x.M_2', & \textit{and} \\ [M_1/x]M_2 \sim [M_1'/x]M_2' \in A_2\,[\,\eta\,] & \textit{if } M_1 \sim M_1' \in A_1\,[\,\eta\,] \end{cases}$$

$$M \sim M' \in \forall X.A_2\,[\,\eta\,] \Leftrightarrow \begin{cases} M \longmapsto^* \Lambda.M_2,\ M' \longmapsto^* \Lambda.M_2', & \textit{and} \\ M_2 \sim M_2' \in A_2\,[\,\eta[X \mapsto R]\,] & \textit{if } R\ \mathsf{Cand} \end{cases}$$

It is immediate that logical similarity is closed under head expansion, given that candidates are required to be.

For η, η' candidate assignments for Δ, let $\eta^{\mathsf{op}}(X) \triangleq \eta(X)^{\mathsf{op}}$ and $(\eta' \circ \eta)(X) \triangleq \eta'(X) \circ \eta(X)$, for each X in Δ.

Lemma 3.2 (Opposition and Composition). *1. $M \sim M' \in A\,[\,\eta\,]$ iff $M' \sim M \in A\,[\,\eta^{\mathsf{op}}\,]$.*

2. If $M \sim M' \in A\,[\,\eta\,]$ and $M' \sim M'' \in A\,[\,\eta'\,]$, then $M \sim M'' \in A\,[\,\eta' \circ \eta\,]$

Proof. Each is proved by induction on the structure of A. □

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

1. $\widehat{\gamma}(M) \sim \widehat{\gamma'}(M) \in A\,[\,\eta\,]$,
2. $\widehat{\gamma}(M') \sim \widehat{\gamma'}(M') \in A\,[\,\eta\,]$, and
3. $\widehat{\gamma}(M) \sim \widehat{\gamma'}(M') \in A\,[\,\eta\,]$,

The first two conditions stipulate that corresponding instances of M, and of M', are similar; the third states that the γ instance of M is similar to the γ' instance of M'.

For brevity, write $\Gamma \gg_\Delta M \in A$ to mean $\Gamma \gg_\Delta M \doteq M \in A$, and note that if $\Gamma \gg_\Delta M \doteq M' \in A$, then $\Gamma \gg_\Delta M \in A$ and $\Gamma \gg_\Delta M' \in A$.

The fundamental theorem is proved as in **PFPL**.

Theorem 3.3 (FTLR). *If $\Gamma \vdash_\Delta M : A$, then $\Gamma \gg_\Delta M \in A$.*

The equational extension of the FTLR states that if $\Gamma \vdash_\Delta M \equiv M' : A$, then $\Gamma \gg_\Delta M \doteq M' \in A$. The proof reduces to a series of lemmas about exact equality.

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

$$\Gamma \gg_\Delta \mathsf{ap}((\,\lambda x.M_2\,), M_1) \doteq [M_1/x]M_2 \in A_2.$$

2. If $\Gamma \gg_{\Delta, X\ \mathsf{type}} M_2 \in A_2$ and $\Delta \vdash A_1\ \mathsf{type}$, 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 $\Gamma \gg_\Delta M \doteq \lambda x.\mathsf{ap}(M, x) \in A_1 \to A_2$.

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 easily obtained. The FTLR states reflexivity for formally well-typed terms.

What about symmetry and transitivity?

1. If $\Gamma \gg_\Delta M \doteq M' \in A$, then $\Gamma \gg_\Delta M' \doteq M \in A$.

2. If $\Gamma \gg_\Delta M \doteq M' \in A$ and $\Gamma \gg_\Delta M' \doteq M'' \in A$, then $\Gamma \gg_\Delta M \doteq M'' \in A$.

These are both proved using the opposition and composition and zig-zag lemmas.

1. Assume the antecedent, let η be a candidate assignment for Δ, and suppose $\gamma' \sim \gamma \in \Gamma\,[\,\eta\,]$. By the opposition lemma $\gamma \sim \gamma' \in \Gamma\,[\,\eta^{\mathsf{op}}\,]$. Noting that η^{op} is a candidate assignment, it follows from the assumption that $\widehat{\gamma}(M) \sim \widehat{\gamma'}(M') \in A\,[\,\eta^{\mathsf{op}}\,]$. But then by the opposition lemma $\widehat{\gamma'}(M') \sim \widehat{\gamma}(M) \in A\,[\,\eta\,]$, as required.

2. Assume the antecedents, let η be a candidate assignment for Δ, and suppose that $\gamma \sim \gamma'' \in A\,[\,\eta\,]$. By the opposition and composition lemma $\gamma \sim \gamma \in A\,[\,\eta^{\mathsf{op}} \circ \eta\,]$, and by the zig-zag lemma, $\eta^{\mathsf{op}} \circ \eta$ is a candidate assignment for Δ. So, by the first assumption, $\widehat{\gamma}(M) \sim \widehat{\gamma}(M') \in A\,[\,\eta^{\mathsf{op}} \circ \eta\,]$, and by the second $\widehat{\gamma}(M') \sim \widehat{\gamma''}(M'') \in A\,[\,\eta\,]$. The composition lemma implies that $\widehat{\gamma}(M) \sim \widehat{\gamma''}(M'') \in A\,[\,\eta \circ \eta^{\mathsf{op}} \circ \eta\,]$. But by zig-zag completeness $\eta \circ \eta^{\mathsf{op}} \circ \eta = \eta$, which completes the proof.

References

Robert L. Constable, Stuart F. Allen, Mark Bromley, Rance Cleaveland, J. F. Cremer, R. W. Harper, Douglas J. Howe, Todd B. Knoblock, N. P. Mendler, Prakash Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing mathematics with the Nuprl proof development system.* Prentice Hall, 1986. ISBN 978-0-13-451832-9. URL `http://dl.acm.org/citation.cfm?id=10510`.

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

Robert Harper. How to (re)invent Girard's method. Supplement to Harper (2016), Spring 2020a. URL https://www.cs.cmu.edu/~rwh/pfpl/supplements/girard.pdf.

Robert Harper. How to (re)invent Tait's method. Supplement to Harper (2016), Spring 2020b. URL https://www.cs.cmu.edu/~rwh/pfpl/supplements/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.

Per Martin-Löf. Constructive mathematics and computer programming. In L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski, editors, *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science, Hannover 1979*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 153–175. North-Holland, 1982. doi: 10.1016/S0049-237X(09)70189-2. URL http://dx.doi.org/10.1016/S0049-237X(09)70189-2.

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.