How to (Re)Invent Girard's Method[*]

Robert Harper

Spring, 2021

1 Introduction

Harper (2024) shows how to reinvent Tait's Method, the starting point for the general theory of logical relations, a central method in type theory. The extension to account for polymorphism is not obvious, the difficulty being that a form of circularity, called *impredicativity*, is inherent in the notion of type quantification. That obstacle was overcome by Girard using what is now known as (wait for it) *Girard's Method*. The purpose of this note is to show how one might reinvent his method by considering again the termination of head reduction for System F of variable types.

2 Polymorphic Types

The syntax of System F is given by the following grammar:

$$
\begin{aligned}
A &::= X \mid \mathsf{ans} \mid A_1 \to A_2 \mid \forall(X.A) \\
M &::= x \mid \mathsf{yes} \mid \mathsf{no} \mid \lambda(x.M) \mid \mathsf{ap}(M_1;M_2) \mid \Lambda(X.M) \mid \mathsf{Ap}(M;A)
\end{aligned}
$$

Here X is a *type variable* and x is a *term variable*, the former ranging over types, the latter over terms of a type.

A *type variable context*, Δ, is a finite set of declarations X_1 type, ... X_n type, and a *term variable context* is a finite set of declarations $x_1 : A_1, \dots, x_n : A_n$, with no variable declared more than once. The statics is given by two judgments, $\Delta \vdash A$ type, stating that A is a well-formed type relative to assumptions Δ, and $\Gamma \vdash_\Delta M : A$, stating that M is of type A under assumptions Δ and Γ. The inductive definitions of these judgments are given in Figure 1. These definitions are arranged so that contraction and exchange (in either context) is implicitly admissible by virtue of considering sets of assumptions, and weakening and substitution may be shown to be admissible by induction on derivations.

Lemma 1. *The following substitution and weakening principles are admissible:*

1. *If $\Delta \vdash A$ type, then Δ, X type $\vdash A$ type.*

2. *If $\Gamma \vdash_\Delta N : B$, then $\Gamma \vdash_{\Delta, X\ type} N : B$.*

3. *If $\Delta \vdash A$ type and Δ, X type $\vdash B$ type, then $\Delta \vdash [A/X]B$ type.*

[*]Copyright © Robert Harper. All Rights Reserved.

$$\frac{\text{VAR}}{\Gamma, x : A \vdash_\Delta x : A} \qquad \frac{\text{YES}}{\Gamma \vdash_\Delta \mathsf{yes} : \mathsf{ans}} \qquad \frac{\text{NO}}{\Gamma \vdash_\Delta \mathsf{no} : \mathsf{ans}}$$

LAM
$$\frac{\Gamma, x : A_1 \vdash_\Delta M_2 : A_2}{\Gamma \vdash_\Delta \lambda(x.M_2) : A_1 \to A_2}$$

APP
$$\frac{\Gamma \vdash_\Delta M_1 : A_2 \to A \qquad \Gamma \vdash_\Delta M_2 : A_2}{\Gamma \vdash_\Delta \mathsf{ap}(M_1;M_2) : A}$$

LAM
$$\frac{\Gamma \vdash_{\Delta, X\,\mathsf{type}} M : A}{\Gamma \vdash_\Delta \Lambda(X.M) : \forall(X.A)}$$

APP
$$\frac{\Gamma \vdash_\Delta M : \forall(X.B) \qquad \Delta \vdash A\,\mathsf{type}}{\Gamma \vdash_\Delta \mathsf{Ap}(M;A) : [A/X]B}$$

Figure 1: Statics of System F

$$\frac{\text{YES}}{\mathsf{yes}\ \mathsf{final}} \qquad \frac{\text{NO}}{\mathsf{no}\ \mathsf{final}}$$

APP
$$\frac{M_1 \longmapsto M_1'}{\mathsf{ap}(M_1;M_2) \longmapsto \mathsf{ap}(M_1';M_2)}$$

APP-LAM
$$\frac{}{\mathsf{ap}(\lambda(x.M);M_2) \longmapsto [M_2/x]M}$$

APP
$$\frac{M \longmapsto M'}{\mathsf{Ap}(M;A) \longmapsto \mathsf{Ap}(M';A)}$$

APP-LAM
$$\frac{}{\mathsf{Ap}(\Lambda(X.M);A) \longmapsto [A/X]M}$$

Figure 2: Dynamics of System F

4. *If* $\Delta \vdash A$ *type and* $\Gamma \vdash_{\Delta, X\ type} N : B$*, then* $[A/X]\Gamma \vdash_\Delta [A/X]N : [A/X]B$.

5. *If* $\Gamma \vdash_\Delta M : A$ *and* $\Gamma, x : A \vdash_\Delta N : B$*, then* $\Gamma \vdash_\Delta [M/x]N : B$.

The dynamics is inductively defined by the rules in Figure 2. It defines weak head reduction for closed terms (those with neither free type variables nor free ordinary variables). The reason to include the special base type ans of answers is to provide a directly observable notion of the outcome of a computation, either yes or no, which can be interpreted as an accept/reject signal. Complete programs are therefore defined to be closed terms of type ans; from a machine-theoretic viewpoint complete programs consist of the program *per se* together with its input (encoded as terms). Note well that types are never evaluated or simplified in any way during execution!

Lemma 2 (Preservation). *If* $\vdash_\emptyset M : A$ *and* $M \longmapsto M'$*, then* $\vdash_\emptyset M' : A$.

3 Termination Proof

The most obvious strategy for proving termination is as a direct generalization of Tait's Method for the simply typed λ-calculus. Because type variables range over types, it is natural to consider all closed instances of types by substitution of closed types for type variables as follows:

Theorem 3. *If $\Gamma \vdash_\Delta M : A$, then for any closing substitution $\delta : \Delta$ for type variables and any hereditarily terminating closing substitution $\gamma : \Gamma$ for term variables, the instance $\hat{\gamma}(\hat{\delta}(M))$ is hereditarily terminating of type $\hat{\delta}(A)$.*

Assuming that hereditary termination at type ans implies termination, the desired result of termination of complete programs follows directly.

This is well and good, provided that an appropriate generalization of hereditary termination can be given that accounts for polymorphic types. The obvious definition builds on the same principle that type variables range over closed types. Thus, a closed term M is hereditarily terminating at type $\forall(X.B)$ iff $\mathsf{Ap}(M;A)$ is hereditarily terminating at type $[A/X]B$ for every closed type A. This seems natural enough, and would suffice for the above theorem, if only hereditary termination were properly defined by this criterion.

It is not.

The trouble is that in the case of Tait's Method hereditary termination at a type A is defined by induction on the structure of A. And here's the rub: a substitution instance $[A/X]B$ of a type B with a free type variable X in it can be *larger* (in any known sense) than B itself, and hence than $\forall(X.B)$. For example, if $A = \forall(X.X \to X)$, then $[A/X](X \to X) = A \to A$, which is strictly larger than A. Not only is it structurally larger, but the instance also has more occurrences of the type quantifier than does the original term, in contrast to the first-order quantifier whose instances have fewer, even though they may be larger in size.

What to do?

One move is to engage in what Lakatos (2015) calls "monster barring," which is to rule out the counterexamples. The standard method is to introduce a form of *type stratification* in which *simple* types do not involve any quantification, whereas *polytypes* extend simple types to include quantified types. The trick is then to demand that type variables range only over simple types, so that $[A/X]B$ is smaller than B whenever A is simple, for the simple reason that the instance has one fewer type quantifier. The language with this restriction is said to be *predicative*, rather than *impredicative*. Restricting to the predicative fragment is enough to salvage Tait's Method more or less intact.

But it does so at the expense of reducing the expressive power of the language. For example, no longer are type constructors, such as product types, definable by their universal properties—exactly because the universal conditions demand for their force on quantification over *all* other types with specified structure, rather than just the small ones. On the other hand, who says these types ought to be *defined* by their universal properties, rather than just being *characterized* by them? Well, no one. And indeed the whole of dependent type theory is based on embracing predicativity, and defining all type constructors independently, and then showing that their universal properties hold. After all, even the impredicative encodings do not achieve universality: the unicity conditions must be imposed by other means, so nothing is truly lost by avoiding the encodings.

Be that as it may, let us press on and consider an alternative to monster barring, namely proving a much stronger property of terms that implies the desired syntactic criterion needed for the main result. The trick is to think of the property "M is hereditarily terminating at type A" as a specification of the *behavior* of M under head reduction. Tait's Method breaks the termination proof into two parts: (1) assigning a specification to a type by structural induction, and (2) showing that a well-typed term behaves according to the induced specification.

Although types induce specifications, there is nothing to say that the *only* specifications are those induced by types. For example, in the presence of a type of natural numbers, the type $\mathsf{nat} \to \mathsf{nat}$ induces the specification “when applied to a computation of a nat, yields a computation of a nat.” And every term of this type satisfies that specification. But one may also consider a specification of functions of this type such as “when applied to a computation of a prime, yields a computation of a perfect square.” This property is well-defined, and some programs even satisfy it, but it is not a specification that is induced by the static type discipline. In fact there are *uncountably many* such specifications laying around in the world, but only countably many of them are induced by types.

Girard’s Method exploits this discrepancy.

It is fine to say that types induce specifications of behavior, but what is a behavioral specification in general? The answer, in the present context, is any property that is *closed under head expansion*, which is to say that if M satisfies the property, and $M' \longmapsto M$, then M' does as well. Although closure under reverse execution may seem like an odd requirement, a moment’s thought reveals that it is simply the expression of a natural condition on specifications, namely that they are determined by how a program *behaves*, rather than on the details of what is *is*. For example, the property of a term stating that it “contains three type quantifiers” is not closed under head expansion, and so is not a valid specification. On the other hand, the prime-to-square property *is* a valid specification, because it speaks only of how a function acts when applied, rather than what it is.

The crucial move made by Girard is to enlarge the range of significance of a type variable from being all closed type expressions, which is circular, to all possible behavioral specifications, which is not. The remarks made earlier about the size of a substitution instance of a type being larger than the type is *irrelevant* to Girard’s method. It achieves this by instead accepting that the collection of “all possible specifications” is well-defined. This might seem innocuous to someone used to accepting all sorts of set constructions, but it is well to understand why it might be considered dubious. Because the termination of System F implies the consistency of a strong logical system called *second-order arithmetic*, Gödel’s Theorem tells us that the termination proof must use methods that go beyond mere behavioral specifications. And, indeed, Girard’s Method does just this, by postulating a set of *all possible specifications*, which in the jargon of the field are called *type candidates*.

It is now possible to outline the technical means by which Girard proves termination for System F.[1]

A *candidate for a closed type A* is a set of closed terms of type A that is closed under head expansion. For each type A it is essential to postulate that the set of all candidates for A is well-defined (even though it is more dubious than the termination property to be proved). Thus, hereditary termination at an type A with free type variables must be defined *relative to* an assignment of type candidates to those type variables.

More precisely, a type substitution δ for Δ, written $\delta : \Delta$, is an assignment of a closed type to each type variable declared in Δ. A candidate assignment η for $\delta : \Delta$, written $\eta \subseteq \delta : \Delta$, is an assignment of a candidate for type $\delta(X)$ to each type variable X declared in Δ. For $\Delta \vdash A$ type, $\delta : \Delta$, and $M : \hat{\delta}(A)$, define M to be *hereditarily terminating at type A relative to* $\eta \subseteq \delta : \Delta$ by induction on the structure of A as follows:

1. If $A = \mathsf{ans}$, then M is hereditarily terminating at type A (rel. $\eta \subseteq \delta : \Delta$) iff M terminates (with either yes or no).

2. If $A = X$, a type variable, then M is hereditarily terminating at type A (rel. $\eta \subseteq \delta : \Delta$) iff $M \in \eta(X) \subseteq \delta(X)$, the candidate assigned to X.

[1]As with Tait’s Method, his actual proof was of strong normalization, but termination of closed terms under weak head reduction is enough to lay bare the critical moves in the game.

3. If $A = A_1 \to A_2$, then M is hereditarily terminating at type A (rel. $\eta \subseteq \delta : \Delta$) iff for all M_1 hereditarily terminating at A_1 (rel. $\eta \subseteq \delta : \Delta$), the application $\mathsf{ap}(M;M_1)$ is hereditarily terminating at A_2 (rel. $\eta \subseteq \delta : \Delta$).

4. If $A = \forall(X.B)$, then M is hereditarily terminating at type A (rel. $\eta \subseteq \delta : \Delta$) iff for all closed types C and all candidates $\mathcal{C}$ for type C, the application $\mathsf{Ap}(M;C)$ is hereditarily terminating at type B (rel. $\eta[X \longmapsto \mathcal{C}] \subseteq \delta[X \longmapsto C] : \Delta, X\ \mathsf{type}$).

Thus the candidate assignment "cuts the knot" by providing an abstract meaning for type variables consistent with their concrete meaning given by a type substitution. Type quantification imposes a strong requirement that a term of this type must be hereditarily terminating for *any* candidate assignment for the quantified type variable, not just the canonical one associated to a given instance.

Lemma 4. *The property of hereditary termination at type A relative to a candidate assignment η is well-defined, and is itself a candidate for A relative to η.*

Proof. Hereditary termination is defined by induction on the structure of types. Because it speaks only of the behavior of terms under head reduction, hereditary termination is itself easily seen to be closed under head expansion, and so is among the candidates considered for interpretation of a type variable. □

With this construction in hand, it is straightforward to prove, *a la* Tait, that every instance of every term inhabits the candidate associated to its type, relative to an assignment of candidates to the free type variables.

For $\Delta \vdash A\ \mathsf{type}$, $\delta : \Delta$, and $\eta \subseteq \delta : \Delta$, write $M \in A\,[\eta]$ to mean that M is hereditarily terminating at type A (rel. $\eta \subseteq \delta : \Delta$). Define $\gamma : \Gamma$ to mean that γ is a substitution of closed terms of type declared Γ, and define $\gamma \in \Gamma\,[\eta]$, where $\gamma : \Gamma$, to mean that for every variable in Γ of type A, if $\gamma(x) = M$, then $M \in A\,[\eta]$. Finally, define $\Gamma \gg_\Delta M \in A$ to mean that if $\delta : \Delta$, $\eta \subseteq \delta : \Delta$, and $\gamma \in \Gamma\,[\eta]$, then $\hat{\gamma}(\hat{\delta}(M)) \in A\,[\eta]$.

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

The notation is a bit daunting, but the theorem merely states that a well-typed term satisfies the behavioral property of hereditary termination. It is an immediate corollary that if $M : \mathsf{ans}$ is a complete program, then either $M \longmapsto^* \mathsf{yes}$ or $M \longmapsto^* \mathsf{no}$, as was to be proved.

4 Proof of Main Theorem

Lemma 6 (Compositionality)**.** *Suppose Δ, X type $\vdash B$ type and $\Delta \vdash A$ type. Then*

$$M \in [A/X]B\,[\eta] \text{ iff } M \in B\,[\eta[X \longmapsto - \in A\,[\eta]]]$$

for any candidate assignment η.

Proof. By induction on the structure of B.

1. $B = X$: Then $[A/X]B = A$.

   By definition $M \in [A/X]B\,[\eta]$ iff $M \in A\,[\eta]$ iff $M \in X\,[\eta[X \longmapsto - \in A\,[\eta]]]$.

2. $B = Y \neq X$: Then $[A/X]B = B$.

   By definition $M \in [A/X]B\,[\eta]$ iff $M \in B\,[\eta]$ iff $M \in B\,[\eta[X \longmapsto - \in A\,[\eta]]]$.

3. $B = B_1 \to B_2$: Then $[A/X]B = [A/X]B_1 \to [A/X]B_2$.

   Suppose $M \in [A/X]B\,[\eta]$ so as to show $M \in B\,[\eta[X \longmapsto - \in A\,[\eta]]]$.

   To this end suppose that
   $$M_1 \in B_1\,[\eta[X \longmapsto - \in A\,[\eta]]],$$
   and show
   $$\mathsf{ap}(M;M_1) \in B_2\,[\eta[X \longmapsto - \in A\,[\eta]]].$$

   By induction $M_1 \in [A/X]B_1\,[\eta]$, and so by assumption $\mathsf{ap}(M;M_1) \in [A/X]B_2\,[\eta]$. But then by induction $\mathsf{ap}(M;M_1) \in B_2\,[\eta[X \longmapsto - \in A\,[\eta]]]$, as required.

   The converse is proved similarly.

4. $B = \forall(Y.B')$ with $Y \neq X$: Then $[A/X]B = \forall(Y.[A/X]B')$, because A is closed.

   Suppose $M \in [A/X]B\,[\eta]$ so as to show $M \in B\,[\eta[X \longmapsto - \in A\,[\eta]]]$. Suppose C type and let $\mathcal{C}$ be a candidate for C, it is enough to show
   $$\mathsf{Ap}(M;C) \in B'\,[\eta[X \longmapsto - \in A\,[\eta]][Y \longmapsto \mathcal{C}]].$$

   By assumption and by the definition of hereditary termination at polymorphic type
   $$\mathsf{Ap}(M;C) \in [A/X]B'\,[\eta[Y \longmapsto \mathcal{C}]].$$

   But then by induction
   $$\mathsf{Ap}(M;C) \in B'\,[\eta[Y \longmapsto \mathcal{C}][X \longmapsto - \in A\,[\eta[Y \longmapsto \mathcal{C}]]]].$$

   Because A cannot involve the type variable Y, the predicates $- \in A\,[\eta[Y \longmapsto \mathcal{C}]]$ and $- \in A\,[\eta]$ are equivalent, which suffices for the result.

   The converse is proved similarly.

□

Exercise 1. *Complete the proof of compositionality for the indicated converse cases omitted in the foregoing incomplete proof.*

In what follows if $\Delta \vdash A$ type, abbreviate $\hat{\delta}(A)$ by $\hat{A}$ whenever $\delta : \Delta$ is evident, and, similarly, if $\Delta \vdash \Gamma$ ctx, write $\hat{\Gamma}$ for $\hat{\delta}(\Gamma)$, the extension of $\hat{\delta}$ to term contexts. If $\Gamma \vdash_\Delta M : A$, write $\hat{M}$ for $\hat{\gamma}(\hat{\delta}(M))$ when $\delta : \Delta$ and $\gamma : \hat{\Gamma}$ are evident.

Proof of main theorem. Proceed by induction on the derivation of $\Gamma \vdash_\Delta M : A$.

1. Rule VAR: Then $\Gamma = \Gamma', x : A$ and $M = x$.

   Suppose $\delta : \Delta$, $\eta \subseteq \delta : \Delta$, $\gamma : \hat{\Gamma}$, and $\gamma \in \Gamma\,[\eta]$. Show that $\hat{\gamma}(\hat{\delta}(M)) \in A\,[\eta]$, which is to say that $\gamma(x) \in A\,[\eta]$.

   This follows immediately from the assumption $\gamma \in \Gamma\,[\eta]$.

2. Rules YES and NO:

   Immediate by definition of hereditary termination at type ans.

3. Rule LAM: Then $M = \lambda(x.M_2)$, $A = A_1 \to A_2$, and $\Gamma, x : A_1 \vdash_\Delta M_2 : A_2$. Note that $\hat{M} = \lambda(x.\hat{M}_2)$, and $\hat{A} = \hat{A}_1 \to \hat{A}_2$.

   Suppose $\delta : \Delta$, $\eta \subseteq \delta : \Delta$, $\gamma : \hat{\Gamma}$, and $\gamma \in \Gamma\,[\eta]$. To show that $\hat{M} \in A\,[\eta]$, suppose $M_1 \in A_1\,[\eta]$ and show that $\mathsf{ap}(\hat{M};M_1) \in A_2\,[\eta]$.

   Let Γ' be $\Gamma, x : A_1$, and let γ' be $\gamma[x \longmapsto M_1]$, so that $\gamma' : \Gamma'$ and $\gamma' \in \Gamma'\,[\eta]$. Note that $\widehat{\gamma'}(\hat{\delta}(M_2)) = [M_1/x]\hat{M}_2$.

   By induction hypothesis $\hat{\gamma'}(M_2) \in A_2\,[\eta]$, which is to say $[M_1/x]\hat{M}_2 \in A_2\,[\eta]$; the result follows by closure under the head expansion

   $$\mathsf{ap}(\hat{M};M_1) \longmapsto [M_1/x]\hat{M}_2.$$

4. Rule APP: Then $M = \mathsf{ap}(M_1;M_2)$, $\Gamma \vdash_\Delta M_1 : A_2 \to A$, and $\Gamma \vdash_\Delta M_2 : A_2$.

   Suppose $\delta : \Delta$, $\eta \subseteq \delta : \Delta$, $\gamma : \hat{\Gamma}$, and $\gamma \in \Gamma\,[\eta]$ to show that $\hat{M} \in A\,[\eta]$.

   By the first inductive hypothesis $\hat{M}_1 \in A_2 \to A\,[\eta]$, and by the second $\hat{M}_2 \in A_2\,[\eta]$.

   The result follows immediately from the definition of hereditary termination at function type.

5. Rule LAM: Then $M = \Lambda(X.N)$, $A = \forall(X.B)$, so that $\hat{M} = \Lambda(X.\hat{N})$ and $\hat{A} = \forall(X.\hat{B})$ (recall that $\delta : \Delta$ is closed), and also $\Gamma \vdash_{\Delta, X\ \mathsf{type}} N : B$.

   To show $\hat{M} \in A\,[\eta]$, it suffices to consider any C type and any candidate $\mathcal{C}$ for C, and show that $\mathsf{Ap}(\hat{M}; C) \in B\,[\eta[X \longmapsto \mathcal{C}]]$.

   This follows easily by induction and closure under the head expansion $\mathsf{Ap}(\Lambda(X.\hat{N}); C) \longmapsto [C/X]\hat{N}$, noting that $\hat{\delta}([C/X]N) = \widehat{\delta[X \longmapsto C]}(N)$ and $\eta[X \longmapsto \mathcal{C}] \subseteq \delta[X \longmapsto C] : \Delta, X\ \mathsf{type}$.

6. Rule APP: Then $M = \mathsf{Ap}(N; C)$, $A = [C/X]B$, $\Gamma \vdash_\Delta N : \forall(X.B)$, and $\Delta \vdash C\ \mathsf{type}$.

   Suppose that $\delta : \Delta$, $\eta \subseteq \delta : \Delta$, $\gamma : \hat{\Gamma}$, and note that $\hat{M} = \mathsf{Ap}(\hat{N}; \hat{C})$, and $\hat{A} = [\hat{C}/X]\hat{B}$, and $\hat{N} : \forall(X.\hat{B})$ and $\hat{C}\ \mathsf{type}$.

   Suppose that $\gamma \in \Gamma\,[\eta]$ so as to show that $\hat{M} \in [C/X]B\,[\eta]$. By compositionality it is enough to show $\hat{M} \in B\,[\eta[X \longmapsto - \in C\,[\eta]]]$.

   It follows from the assumptions that

   $$\eta[X \longmapsto - \in C\,[\eta]] \subseteq \delta[X \longmapsto \hat{C}] : \Delta, X\ \mathsf{type}$$

   because $- \in C\,[\eta]$ is a candidate for type $\hat{C}$.

   But then by the inductive hypothesis $\hat{N} \in \forall(X.B)\,[\eta]$, and so by the definition of hereditary termination at quantified type

   $$\mathsf{Ap}(\hat{N}; \hat{C}) \in B\,[\eta[X \longmapsto - \in C\,[\eta]]],$$

   which was to be shown.

□

Corollary 7. *If $M : A$ is a closed, well-typed term of System F, then M terminates with a value. In particular, if $A = \mathsf{ans}$, then either $M \longmapsto^{*} \mathsf{yes}$ or $M \longmapsto^{*} \mathsf{no}$.*

References

Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

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`.

I. Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge Philosophy Classics. Cambridge University Press, 2015. ISBN 9781316425336. URL `https://books.google.com/books?id=zb8qDgAAQBAJ`. J. Worrall and E. Zahar, eds.

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`.