How to (Re)Invent Tait's Method*

Robert Harper

Spring 2024

1 Introduction

Two of the most important developments in type theory were the invention, by W. W. Tait, of *Tait's Method* for function types, which was later extended by J.-Y. Girard to *Girard's Method* for type quantification, both of which were incorporated into a general theory of *logical relations* for a wide range of type theories. Tait's method continues to be known by its original name, the *computability method*, which interprets types as predicates in the manner developed below.[1]

The problem considered by Tait was to prove that β-reduction for the simply typed λ-calculus is *strongly normalizing*, which is usually defined to mean that there are *no* infinite β-reduction sequences starting with a well-typed term. The question considered here is related, but technically much simpler, the termination of a deterministic head reduction strategy for a simply typed λ-calculus. The type system considered here has unit, product, and function types, augmented with a type of *answers*, yes or no, corresponding to the accept or reject distinction for abstract machines.

2 Simple Types

The syntax of the language considered here is given by the following grammar:

$$\begin{aligned} A &::= 1 \mid \mathsf{ans} \mid A_1 \times A_2 \mid A_1 \to A_2 \\ M &::= x \mid \mathsf{yes} \mid \mathsf{no} \mid \langle\rangle \mid \langle M_1, M_2\rangle \mid M \cdot 1 \mid M \cdot 2 \mid \lambda(x.M) \mid \mathsf{ap}(M_1;M_2) \end{aligned}$$

The statics is entirely standard, defining the typing judgment $\Gamma \vdash M : A$, in such a way that the structural properties are admissible. Contraction and exchange are accounted for by treating the typing context Γ as a finite set of variable typings $x_1 : A_1, \dots, x_n : A_n$ in which $x_i \neq x_j$ whenever $i \neq j$. Weakening is built-in by stating all rules with an ambient typing context Γ that goes along for the ride. See Figure 1 for the definition of typing. Substitution (transitivity), which states that if $\Gamma, x : A \vdash N : B$, and $\Gamma \vdash M : A$, then $\Gamma \vdash [M/x]N : B$, is surprisingly difficult to prove.

Define $\Gamma' \vdash \gamma : \Gamma$ to mean that γ is a finite function defined on variables declared in Γ such that if $\Gamma \vdash x : A$, then $\Gamma' \vdash \gamma(x) : A$. Such a mapping determines a substitution function, $\hat{\gamma}$, on terms that replaces each such x with $\gamma(x)$ throughout the term.

Lemma 1 (Substitution)**.** *If* $\Gamma \vdash M : A$ *and* $\Gamma' \vdash \gamma : \Gamma$*, then* $\Gamma' \vdash \hat{\gamma}M : A$*.*

[1]It can be confusing, at first, that Tait's notion of computability has nothing to do with computability theory!

$$\text{VAR}\ \frac{}{\Gamma, x{:}A \vdash x : A} \qquad \text{YES}\ \frac{}{\Gamma \vdash \mathsf{yes} : \mathsf{ans}} \qquad \text{NO}\ \frac{}{\Gamma \vdash \mathsf{no} : \mathsf{ans}} \qquad \text{UNIT}\ \frac{}{\Gamma \vdash \langle\rangle : 1}$$

$$\text{PAIR}\ \frac{\Gamma \vdash M_1 : A_1 \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \langle M_1, M_2 \rangle : A_1 \times A_2} \qquad \text{LFT}\ \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash M \cdot 1 : A_1} \qquad \text{RHT}\ \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash M \cdot 2 : A_2}$$

$$\text{LAM}\ \frac{\Gamma, x{:}A_1 \vdash M_2 : A_2}{\Gamma \vdash \lambda(x.M_2) : A_1 \to A_2} \qquad \text{APP}\ \frac{\Gamma \vdash M_1 : A_2 \to A \qquad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \mathsf{ap}(M_1;M_2) : A}$$

Figure 1: Typed λ-Calculus Statics

The *structural properties* of typing follow immediately from substitution and the definition of typing:

1. Reflexivity: $x : A \vdash x : A$. This is an instance of the reflexivity rule.
2. Transitivity: If $\Gamma, x{:}A \vdash N : B$, then $\Gamma \vdash M : A$ implies $\Gamma \vdash [M/x]N : B$.
3. Weakening: If $\Gamma \vdash N : B$, then $\Gamma, x{:}A \vdash N : B$, where x is not declared in Γ.
4. Contraction: If $\Gamma, x_1 : A, x_2 : A \vdash N : B$,then $\Gamma, x : A \vdash [x, x/x_1, x_2]N : B$.

Exercise 1 (Difficult)**.** *Attempt to prove Lemma 1 by induction on the derivation of* $\Gamma \vdash M : A$*. There is one spot where your proof is sure to break down. Isolate that spot, and consider other ways to recover the intended result.*

The dynamics is given by a transition system $M \longmapsto M'$ between closed λ-terms of some type. Any closed typed term is a valid initial state. Final states are defined along with transition in Figure 2.

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

Proof. By induction on transition. □

3 Termination Proof

The goal is to prove termination for terms of observable type:

Theorem 3 (Termination)**.** *If* $M : \mathsf{ans}$*, then either* $M \longmapsto^* \mathsf{yes}$ *or* $M \longmapsto^* \mathsf{no}$*.*

That is, any complete program either accepts or rejects.

Given the statement of the theorem, practically the only move available is to proceed by induction on typing. Let us consider some cases.

VAR Does not apply to closed terms.

$$\frac{}{\textsf{yes final}}\ \textsc{yes} \qquad \frac{}{\textsf{no final}}\ \textsc{no} \qquad \frac{}{\langle\rangle\ \textsf{final}}\ \textsc{unit} \qquad \frac{}{\langle M_1, M_2\rangle\ \textsf{final}}\ \textsc{pair} \qquad \frac{}{\lambda(x.M_2)\ \textsf{final}}\ \textsc{lam} \qquad \frac{M \longmapsto M'}{M \cdot 1 \longmapsto M' \cdot 1}\ \textsc{lft}$$

$$\frac{M \longmapsto M'}{M \cdot 2 \longmapsto M' \cdot 2}\ \textsc{rht} \qquad \frac{}{\langle M_1, M_2\rangle \cdot 1 \longmapsto M_1}\ \textsc{lft-pair} \qquad \frac{}{\langle M_1, M_2\rangle \cdot 2 \longmapsto M_2}\ \textsc{rht-pair} \qquad \frac{M_1 \longmapsto M_1'}{\textsf{ap}(M_1;M_2) \longmapsto \textsf{ap}(M_1';M_2)}\ \textsc{app}$$

$$\frac{}{\textsf{ap}(\lambda(x.M);M_2) \longmapsto [M_2/x]M}\ \textsc{app-lam}$$

Figure 2: Typed λ-Calculus Dynamics

YES Immediate, as yes final.

NO Immediate, as no final.

UNIT Does not apply, not of type ans.

PAIR Does not apply, not of type ans.

LFT By induction, *um* ....

RHT By induction, *um* ....

LAM Does not apply, not of type ans.

APP By induction applied to the first premise, *um* ....

All cases are trivial, or completely unclear.

Well, because the subterms of a term of type ans need not have type ans, it seems clear that it is necessary to strengthen the theorem to say something about terms of any type.

Lemma 4. *If $M : A$, then there exists N such that N final and $M \longmapsto^* N$.*

The lemma suffices for the theorem because of the definition of finality for terms of type ans. Let us consider the proof of this lemma.

VAR Does not apply to closed terms.

YES Immediate, as yes final.

NO Immediate, as no final.

UNIT Immediate, as $\langle\rangle$ final.

PAIR Immediate, as $\langle M_1, M_2\rangle$ final.

LFT By induction there exists N such that N final and $M \longmapsto^* N$. By preservation and the definition of finality N must be of the form $\langle N_1, N_2\rangle$. By the definition of transition

$$M \cdot 1 \longmapsto^* \langle N_1, N_2\rangle \cdot 1 \longmapsto N_1.$$

But now what?

RHT Analogous, what to do with N_2?

LAM Immediate, as $\lambda(x.M_2)$ final.

APP By induction applied to the first premise there exists N_1 such that N_1 final and $M_1 \longmapsto^* N_1$. By preservation and the definition of finality N_1 must have the form $\lambda(x.M)$. By the definition of transition

$$\mathsf{ap}(M_1;M_2) \longmapsto^* \mathsf{ap}(\lambda(x.M);M_2) \longmapsto [M_2/x]M.$$

But now what?

In the projection cases the components of the pair are general terms about which nothing is known. In the application case the value of the first argument is a λ-abstraction whose body is an open term (with free variable x) about which nothing is known. This suggests strengthening the lemma by proving a property called *hereditary termination*, which is stronger than mere termination. It should have the following characteristics in order to push through the proof of the strengthened lemma below:

1. A hereditarily terminating expression of type 1 should be terminating, and hence transition to $\langle\rangle$.

2. A hereditarily terminating expression of type ans should be terminating, and hence transition to either yes or no.

3. A hereditarily terminating expression of type $A_1 \times A_2$ should terminate with a pair $\langle N_1, N_2\rangle$ such that both N_1 and N_2 are hereditarily terminating.

4. A hereditarily terminating expression of type $A_2 \to A$ should terminate with a function $\lambda(x.M)$ such that if M_2 is hereditarily terminating of type A_2, then $[M_2/x]M$ should be hereditarily terminating at type A.

These conditions constitute a *definition* of the property *M is hereditarily terminating at type A*, which is defined for closed $M : A$. The first two cases are given outright; the others rely on hereditary termination at constituent types of a compound type. *Thus, hereditary termination at a type is defined by induction on the structure of the type.*[2]

Lemma 5. *If $M : A$, then M is hereditarily terminating at type A.*

[2]For reference the type-indexed family of predicates, $\mathsf{HT}_A(M)$, defining hereditary termination is given in Figure 3.

$$\mathsf{HT}_1(M) \text{ iff } M \longmapsto^* \langle\rangle$$

$$\mathsf{HT}_{\mathsf{ans}}(M) \text{ iff } M \longmapsto^* \mathsf{yes} \text{ or } M \longmapsto^* \mathsf{no}$$

$$\mathsf{HT}_{A_1\times A_2}(M) \text{ iff } M \longmapsto^* \langle M_1, M_2\rangle \text{ and } \mathsf{HT}_{A_1}(M_1) \text{ and } \mathsf{HT}_{A_2}(M_2)$$

$$\mathsf{HT}_{A_1\to A_2}(M) \text{ iff } M \longmapsto^* \lambda(x.M_2) \text{ and } \mathsf{HT}_{A_1}(M_1) \text{ implies } \mathsf{HT}_{A_2}([M_1/x]M_2)$$

$$\mathsf{HT}_\Gamma(\gamma) \text{ iff } \mathsf{HT}_A(\gamma(x)) \text{ for all } x : A \in \Gamma$$

Figure 3: Hereditary Termination, $\mathsf{HT}_A(M)$

The proof proceeds as before by induction on typing. The cases for the constants are immediate by the definition of hereditary termination at base type.

The problematic elimination cases use the definition of hereditary termination, along with an additional property, called *head expansion*. Before stating it, let us see how it arises. Consider the rule LFT once again. By induction on the premise of the rule, $\mathsf{HT}_{A_1\times A_2}(M)$. By the definition of hereditary termination $M \longmapsto^* \langle M_1, M_2\rangle$ and $\mathsf{HT}_{A_1}(M_1)$. To show $\mathsf{HT}_{A_1}(M \cdot 1)$, observe that

$$M \cdot 1 \longmapsto^* \langle M_1, M_2\rangle \cdot 1 \longmapsto M_1.$$

To complete the proof it suffices to show that hereditary termination is closed under "reverse execution".

Lemma 6 (Head Expansion). *If $HT_A(M)$ and $M' \longmapsto M$, then $HT_A(M')$.*

Proof. Immediate, because the definition of hereditary termination is defined in terms of the evaluation behavior of terms. □

This completes the proof for the rule LFT; rules RHT and APP are handled similarly.

What about the pair and function cases?

PAIR By induction M_1 is hereditarily terminating at A_1 and M_2 is hereditarily terminating at type A_2; the goal is to show that $\langle M_1, M_2\rangle$ is hereditarily terminating at type $A_1 \times A_2$. A pair is already a value (final state), so an appeal to the inductive hypothesis suffices to finish the proof.

LAM To show that $\lambda(x.M_2)$ is hereditarily terminating at $A_1 \to A_2$, show that whenever M_1 is hereditarily terminating at A_1, then $[M_1/x]M_2$ is hereditarily terminating at A_2. But what to do?

The problem now is that in the function case there is no inductive hypothesis available to give us the necessary information about the *open* term M, which has one free variable, x, in it. The lemma must be strengthened once more to account for open terms, even though the desired property applies only to closed terms.

The judgment $\Gamma \gg M \in A$ is defined to mean that if $\mathsf{HT}_\Gamma(\gamma)$, then $\mathsf{HT}_A(\hat{\gamma}(M))$. We may then state the fundamental theorem as follows:

Theorem 7. *If* $\Gamma \vdash M : A$*, then* $\Gamma \gg M \in A$*.*

Proof. The proof is by induction on typing derivations. The critical case is the last one in the preceding attempt, for which the strengthened statement provides precisely what is needed to push the proof through. The other cases require a bit more care in handling the application of γ to the terms in question, but there are no further obstacles to the proof. $\square$

And that is Tait's Method!

Exercise 2. *If termination is required only for closed programs of answer type, and not for higher types, then a "negative" formulation of hereditary termination is sensible:*

$$\mathit{HT}_{A_1 \times A_2}(M) \text{ iff } \mathit{HT}_{A_1}(M \cdot 1) \text{ and } \mathit{HT}_{A_2}(M \cdot 2)$$
$$\mathit{HT}_{A_1 \to A_2}(M) \text{ iff } \mathit{HT}_{A_1}(M_1) \text{ implies } \mathit{HT}_{A_2}(\mathit{ap}(M;M_1))$$

Re-prove the termination theorem using this revised definition of hereditary termination at product and function types.

Exercise 3. *Finite sums, given by the empty type* 0 *and the binary sum,* $A_1 + A_2$*, require a "positive" formulation of hereditary termination:*

$$\mathit{HT}_0(M) \text{ iff (never)}$$
$$\mathit{HT}_{A_1 + A_2}(M) \text{ iff } M \longmapsto^* 1 \cdot M_1 \text{ and } \mathit{HT}_{A_1}(M_1), \text{ or}$$
$$M \longmapsto^* 2 \cdot M_2 \text{ and } \mathit{HT}_{A_2}(M_2).$$

Extend the proof of termination to account for sum types based on these definitions. What would be a "negative" formulation of sum types? What goes wrong?

For the next two exercises the typing and transition rules are given in Figure 4.

Exercise 4. *Extend the termination proof to account for the type* nat *of natural numbers, generated by zero and successor, and interpreted by iteration, under a lazy dynamics whereby any successor is a value, regardless of the form of the predecessor. Define hereditary termination at type* nat *as the strongest property* $\mathcal{P}$ *of* M : nat *such that*

1. *If* $M \longmapsto^*$ zero*, then* $\mathcal{P}(M)$*, and*

2. *If* $M \longmapsto^*$ succ(N) *and* $\mathcal{P}(N)$*, then* $\mathcal{P}(M)$*.*

From this definition derive a suitable induction principle to use in the proof of termination by Tait's method.

Exercise 5. *Extend the termination proof to account for the type* conat *of co-natural numbers, which may be tested for zero and successor, and introduced by a generator with internal state of arbitary type. Define hereditary termination at type* conat *to be the weakest property* $\mathcal{P}$ *of* M : conat *such that if* $\mathcal{P}(M)$*, then either*

1. pred$(M) \longmapsto^* 1 \cdot \langle\rangle$*, or*

2. pred$(M) \longmapsto^* 2 \cdot N$ *with* $\mathcal{P}(N)$*.*

From this definition derive a suitable coinduction principle using Tait's method, and use this to prove the fundamental theorem for the type conat*.*

NAT-I-Z

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathsf{nat}}$$

NAT-I-S

$$\frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash \mathsf{succ}(M) : \mathsf{nat}}$$

NAT-E

$$\frac{\Gamma \vdash M : \mathsf{nat} \qquad \Gamma \vdash M_0 : A \qquad \Gamma, x : A \vdash M_1 : A}{\Gamma \vdash \mathsf{natit}\ M \{ M_0 \mid x.M_1 \} : A}$$

ZERO-VAL

$$\frac{}{\mathsf{zero\ val}}$$

SUCC-VAL

$$\frac{}{\mathsf{succ}(M)\,\mathsf{val}}$$

REC-STEP

$$\frac{M \longmapsto M'}{\mathsf{natit}\ M \{ M_0 \mid x.M_1 \} \longmapsto \mathsf{natit}\ M_0 \{ x \mid M_1 . \}}$$

REC-STEP-Z

$$\frac{}{\mathsf{natit\ zero} \{ M_0 \mid x.M_1 \} \longmapsto M_0}$$

REC-STEP-S

$$\frac{}{\mathsf{natit\ succ}(M) \{ M_0 \mid x.M_1 \} \longmapsto [\mathsf{natit}\ M \{ M_0 \mid x.M_1 \}/x] M_1}$$

CONAT-E

$$\frac{\Gamma \vdash M : \mathsf{conat}}{\Gamma \vdash \mathsf{pred}(M) : 1 + \mathsf{conat}}$$

CONAT-I

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x : A \vdash N : 1 + A}{\Gamma \vdash \mathsf{gen}(M; x.N) : \mathsf{conat}}$$

PRED-STEP

$$\frac{M \longmapsto M'}{\mathsf{pred}(M) \longmapsto \mathsf{pred}(M')}$$

PRED-GEN

$$\frac{}{\mathsf{pred}(\mathsf{gen}(M; x.N)) \longmapsto \mathsf{case}\, [M/x] N \{ \_{}_1 .1 \cdot \langle\rangle_1 \mid \_{}_2 .1 \cdot \langle\rangle_2 \} y \mathsf{gen}(y; x.N)}$$

Figure 4: Natural and Co-Natural Numbers

References

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

Robert Harper. Tarski's fixed point theorem. (Unpublished lecture note), Spring 2024. URL `https://www.cs.cmu.edu/~rwh/courses/atpl/notes/tarski.pdf`.

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`.