

Submitted to LICS'89 on Oct 28, 1988

Submitted 10/28

Rejected Dec. 20, 1988

# On a Formal Correspondence Between $\Lambda$ - $\mathcal{C}$ -Terms and Classical Proofs

— Extended Abstract —

Timothy G. Griffin  
Department of Computer Science  
Rice University  
Houston, TX 77251-1892

October 28, 1988

## 1 Introduction

The propositions-as-types correspondence [How80] relates proofs in *constructive* logic to functional programs. The correspondence has intrigued those interested in the formal verification of programs with the possibility of developing programs from proofs of their specifications [BC85, Con86, Moh86]. However, one drawback of this approach to program development is that the programming languages involved are purely functional. These languages cannot express constructs for control of evaluation and for manipulation of state that are so important in practical programming languages.

Functional languages can be extended with constructs for control and state. For example, the  $\Lambda$ - $\mathcal{C}$ -calculus of Felleisen *et al* [FFKD86, FFKD87] is a theory for reasoning about a functional language with *control* constructs. The  $\Lambda$ - $\mathcal{C}$ -calculus extends Plotkin's  $\Lambda_v$ -calculus [Pl075] with two control constructs,  $\mathcal{A}$  and  $\mathcal{C}$ . Roughly speaking,  $\mathcal{A}$  represents an *abort* operation that stops a program and returns with the value of its argument. The operator  $\mathcal{C}$  applies its argument to the current continuation, an abstraction the rest of the computation; it is closely related to the *call/cc* construct in Scheme and to the *catch/throw* mechanism of Lisp.

This paper presents some preliminary results of an attempt to extend the *propositions-as-types* correspondence to  $\Lambda$ - $\mathcal{C}$ -programs. The well-known correspondence between natural deduction proofs and  $\lambda$ -terms [How80, Ste72]

is extended in such a way that proofs are mapped to  $\Lambda$ - $\mathcal{C}$ -terms. What is surprising about the mapping is that it takes *classical proofs* to  $\Lambda$ - $\mathcal{C}$ -terms. The mapping takes instances of the *falsum* rule to  $\mathcal{A}$ -applications and instances of the *reductio ad absurdum* rule to  $\mathcal{C}$ -applications. Reduction rules for classical proofs are defined that correspond to the reduction of associated  $\Lambda$ - $\mathcal{C}$ -terms.

Given this correspondence between classical proofs and  $\Lambda$ - $\mathcal{C}$ -terms, it is natural to view certain transformations of  $\Lambda$ - $\mathcal{C}$ -terms as defining proof transformations. One such transformation is Fischer's [Fis72] *continuation passing style* (cps) transform defined on  $\Lambda_v$ -terms. This transform was extended to  $\Lambda$ - $\mathcal{C}$ -terms by Felleisen *et al* [FFKD86]. Meyer and Wand pointed out that Fischer's transform can be defined on simply typed terms. This observation is recast below as a transformation on proofs. This proof transformation is then extended to classical proofs in such a way that the extended cps transform closely corresponds to a transformation of *classical* proofs into *intuitionistic* proofs.

The abstract is organized as follows. Section 2 provides a brief description of the  $\Lambda$ - $\mathcal{C}$ -calculus of Felleisen *et al*. Section 3 defines a mapping from classical, natural deduction proofs to  $\Lambda$ - $\mathcal{C}$  terms and defines reductions on proofs that correspond to the reduction rules of  $\Lambda$ - $\mathcal{C}$ -terms. Section 4 demonstrates that the cps transform is related to a translation of classical proofs to intuitionistic ones. Finally, Section 5 discusses ongoing research.

## 2 The $\Lambda$ - $\mathcal{C}$ -Calculus

$\Lambda$ - $\mathcal{C}$ -terms are defined with the syntax

$$M := x \mid MN \mid \lambda x.M \mid \mathcal{A}M \mid \mathcal{C}M.$$

The notation  $M[N/x]$  denotes the capture-avoiding substitution of  $N$  for all free  $x$  in  $M$ .

Define a *value* to be either a variable or an abstraction. The metavariables  $V, V_1, V_2, \dots$  will represent values. The following reduction rules for

$\Lambda$ - $\mathcal{C}$ -terms are defined in [FFKD86]:

$$\begin{array}{lll}
(\lambda x.M)V & \xrightarrow{\beta_v} & M[V/x] & (\beta_v) \\
(\mathcal{A}M)N & \xrightarrow{\mathcal{A}_L} & \mathcal{A}M & (\mathcal{A}_L) \\
V(\mathcal{A}N) & \xrightarrow{\mathcal{A}_R} & \mathcal{A}N & (\mathcal{A}_R) \\
(\mathcal{C}M)N & \xrightarrow{\mathcal{C}_L} & \mathcal{C}\lambda k.M(\lambda f.k(fN)) & (\mathcal{C}_L) \\
V(\mathcal{C}N) & \xrightarrow{\mathcal{C}_R} & \mathcal{C}\lambda k.N(\lambda v.k(Vv)) & (\mathcal{C}_R)
\end{array}$$

Let  $\longrightarrow_m$  be the compatible closure of the  $\beta_v$  rule;  $\longrightarrow_i$  the compatible closure of the  $\beta_v$ ,  $\mathcal{A}_L$ , and  $\mathcal{A}_R$  rules; and  $\longrightarrow_c$  be the compatible closure of the union of all five rules. With  $s$  being either  $m$ ,  $i$ , or  $c$ , let  $\longrightarrow_s$  be the reflexive, transitive closure of  $\longrightarrow_s$ , and  $=_s$  be the equivalence relation generated by  $\longrightarrow_s$ .

**Lemma 1** For  $s \in \{m, i, c\}$ , the relation  $\longrightarrow_s$  is Church-Rosser.

**Proof** For  $\longrightarrow_m$ , see Plotkin [Plo75], see Felleisen *et al* [FFKD87] for the others.  $\square$

### 3 Classical Proofs and $\Lambda$ - $\mathcal{C}$ -Terms

Formulas  $A$  are defined as

$$A := \perp \mid P \mid A \rightarrow B$$

where the the  $P \in \mathcal{P}$  are the atomic formula and the formula  $\perp$  represents “false.” Negation is defined in the usual way as  $\neg A = A \rightarrow \perp$ .

Formal derivations  $\Sigma$  are generated using the following rules of natural deduction:

$$\begin{array}{ll}
\begin{array}{c} [A] \\ \vdots \\ B \\ \hline A \rightarrow B \end{array} & (\rightarrow I) \qquad \frac{A \rightarrow B \quad A}{B} \quad (\rightarrow E) \\
\frac{\perp}{A} & (\perp_i) \qquad \frac{\neg\neg A}{A} \quad (\perp_c)
\end{array}$$

Let  $\mathbf{M}$  be the system generated by the rules  $\rightarrow I$  and  $\rightarrow E$ ,  $\mathbf{I}$  be the system  $\mathbf{M}$  extended with the  $\perp_i$  rule, and  $\mathbf{C}$  be the system  $\mathbf{I}$  extended with then  $\perp_c$  rule. A derivation  $\Sigma$  with conclusion  $A$  will often be written as  $\frac{\Sigma}{A}$ .

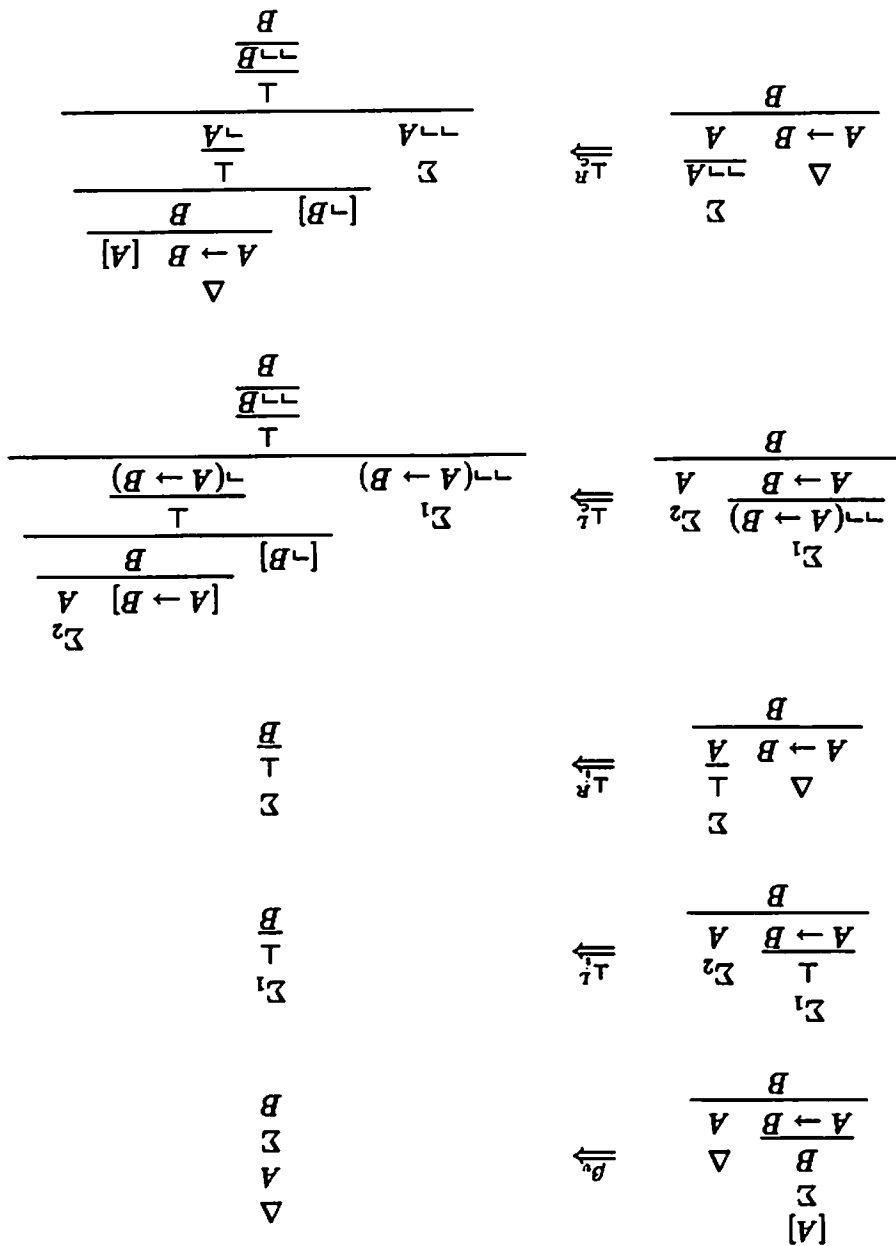
There is a well-known correspondence between natural deduction derivations in  $\mathbf{M}$  and pure  $\lambda$ -terms [How80, Ste72]. This correspondence is extended here by defining a function that maps a  $\mathbf{C}$ -derivation  $\Sigma$  to a  $\Lambda$ - $\mathbf{C}$ -term.

**Definition 1** ( $\tilde{\Lambda}(\Sigma)$ ) *The  $\Lambda$ - $\mathbf{C}$ -term  $\tilde{\Lambda}(\Sigma)$  is defined by induction on  $\Sigma$ .*

1.  $\tilde{\Lambda}(A) = x^A$ , for any formula  $A$ ,
2. if  $\tilde{\Lambda}\left(\frac{\Sigma_1}{A \rightarrow B}\right) = M$  and  $\tilde{\Lambda}\left(\frac{\Sigma_2}{A}\right) = N$ ,  
then  $\tilde{\Lambda}\left(\frac{\frac{\Sigma_1}{A \rightarrow B} \quad \Sigma_2}{B}\right) = MN$ ,
3. if  $\tilde{\Lambda}\left(\frac{A}{\Sigma \quad B}\right) = M$  then  $\tilde{\Lambda}\left(\frac{\frac{[A]}{\Sigma} \quad B}{A \rightarrow B}\right) = \lambda x^A.M$
4. If  $\tilde{\Lambda}\left(\frac{\Sigma}{\perp}\right) = M$ , then  $\tilde{\Lambda}\left(\frac{\Sigma}{\frac{\perp}{A}}\right) = AM$ ,
5. if  $\tilde{\Lambda}\left(\frac{\Sigma}{\neg\neg A}\right) = M$  then  $\tilde{\Lambda}\left(\frac{\Sigma}{\frac{\neg\neg A}{A}}\right) = CM$ .

The variables  $x^A$  of this definition are understood to be untyped variables that have been tagged with a formula. This extends the propositions-as-types principle of Howard in such a way that if  $M = \tilde{\Lambda}\left(\frac{\Sigma}{A}\right)$  then  $M$  can be said to have type  $A$ . Note that a given  $\Lambda$ - $\mathbf{C}$ -term  $M$  may have many types. For example,  $\lambda x.Ax = \tilde{\Lambda}\left(\frac{\perp}{\frac{\perp}{\perp \rightarrow A}}\right)$  for any formula  $A$ .

Figure 1: Reduction rules for C-derivations.



Next, define reduction relations,  $\Rightarrow_s$ , for  $s \in \{m, i, c\}$ , on derivations such that if  $\Sigma_1 \Rightarrow_s \Sigma_2$ , then  $\tilde{\Lambda}(\Sigma_1) \rightarrow_s \tilde{\Lambda}(\Sigma_2)$ . First, a notion corresponding to *value* needs to be defined for derivations.

**Definition 2** A derivation  $\frac{\Sigma}{A}$  is a value derivation if  $\Sigma = A$ , that is, an assumption, or if  $A = B \rightarrow C$  and  $\rightarrow I$  is the last rule application in  $\Sigma$ .

Clearly, if  $\Sigma$  is a value-derivation, then  $\tilde{\Lambda}(\Sigma)$  is a value. The metavariables  $\Delta, \Delta_1, \Delta_2, \dots$  will represent value-derivations.

Figure 1 presents the reduction rules for C-derivations corresponding to the reductions of  $\Lambda$ -C-terms. Let  $\Rightarrow_m$  be the compatible closure of the  $\xRightarrow{\beta_v}$  rule;  $\Rightarrow_i$  the compatible closure of the  $\xRightarrow{\beta_v}$ ,  $\xRightarrow{\beta_i^L}$ , and  $\xRightarrow{\beta_i^R}$  rules;  $\Rightarrow_c$  the compatible closure of the union of all five rules of Figure 1. With  $s$  being either  $m, i$ , or  $c$ , let  $\Rightarrow_s$  be the reflexive, transitive closure of  $\Rightarrow_s$ , and  $=_s$  be the equivalence relation generated by  $\Rightarrow_s$ .

**Theorem 2** The relation  $\Rightarrow_s$  is Church-Rosser, where  $s \in \{m, i, c\}$ .

**Proof Sketch.** The proof follows from Lemma 1 and the following facts:

1.  $\Sigma_2 \Rightarrow_s \Sigma_1$  implies  $\tilde{\Lambda}(\Sigma_2) \rightarrow_s \tilde{\Lambda}(\Sigma_1)$ .
2. If  $M_1 = \tilde{\Lambda}(\Sigma_1)$  and  $M_1 \rightarrow_s M_2$ , then there exists a  $\Sigma_2$  such that  $\Sigma_1 \Rightarrow_s \Sigma_2$  and  $M_2 = \tilde{\Lambda}(\Sigma_2)$ .

□

## 4 The CPS Transform as a Proof Transform

The continuation transform  $\overline{M}$  was introduced by Fischer [Fis72]:

$$\begin{aligned} \overline{x} &= \lambda k.kx \\ \overline{\lambda x.M} &= \lambda k.k(\lambda x.\overline{M}) \\ \overline{MN} &= \lambda k.\overline{M}(\lambda m.\overline{N}(\lambda n.mnk)) \end{aligned}$$

It was extended to  $\Lambda$ -C-terms by Felleisen *et al* [FFKD86]:

$$\begin{aligned} \overline{AM} &= \lambda k.\overline{M}J \\ \overline{CM} &= \lambda k.\overline{M}(\lambda m.m(\lambda v.\lambda k'.kv)J) \end{aligned}$$

where  $J$  was defined to be  $\lambda x.x$ .

Although this cps transform is defined for the untyped  $\Lambda$ -terms, Meyer and Wand [MW85] have shown that Fischer's transform can be seen as defining a transform of simply-typed terms. Assuming there is one distinguished formula (type)  $\mathcal{O}$ , define the transformation  $A^*$  on formulae (types).

$$\begin{aligned} \mathcal{O}^* &= \mathcal{O} \\ P^* &= P \\ (A \rightarrow B)^* &= A^* \rightarrow (B^* \rightarrow \mathcal{O}) \rightarrow \mathcal{O}. \end{aligned}$$

Then, if  $M$  is a pure  $\lambda$ -term that can be assigned type  $A$ ,  $\overline{M}$  can be assigned type  $(A^* \rightarrow \mathcal{O}) \rightarrow \mathcal{O}$ .

This section recasts the Meyer/Wand observation as a transformation on derivations. The proof transformation is extended to  $\mathbf{C}$ -derivations using the extended cps transform where  $J$  is defined to be  $\lambda x.Ax$ .

For  $\mathbf{S}$  being either  $\mathbf{M}$ ,  $\mathbf{I}$ , or  $\mathbf{C}$ , let  $\Gamma \vdash_{\mathbf{S}} A$  represent the assertion that there exists an  $\mathbf{S}$ -derivation for  $A$ , all of whose undischarged assumptions are in the set of formulae  $\Gamma$ . Let  $\Gamma^* = \{A^* \mid A \in \Gamma\}$ . It can now be seen that the cps transform corresponds to a transformation taking derivations in  $\mathbf{C}$  to derivations in  $\mathbf{I}$ .

**Theorem 3** *If there is a proof  $\Sigma$  of  $\Gamma \vdash_{\mathbf{C}} A$  then there exists a proof  $\Sigma^*$  of  $\Gamma^* \vdash_{\mathbf{I}} (A^* \rightarrow \mathcal{O}) \rightarrow \mathcal{O}$  such that*

$$\overline{\tilde{\Lambda}(\Sigma)} = \tilde{\Lambda}(\Sigma^*).$$

**Proof.** By induction on  $\Sigma$ .

Case 1: If  $\Sigma = A$  then let  $\Sigma^* = \frac{A^* \quad [A^* \rightarrow \mathcal{O}]}{\mathcal{O} \rightarrow \mathcal{O}}$ .

Case 2: If  $\Sigma =$

$$\frac{\begin{array}{c} \Sigma_1 \quad \Sigma_2 \\ C \rightarrow A \quad C \end{array}}{A}$$

$$\frac{\frac{(A \rightarrow O) \rightarrow O}{O}}{(T \rightarrow O) \rightarrow O} \quad \Sigma_1$$

Case 4: If  $\Sigma = \frac{A}{T}$  then let  $\Sigma^* = \Sigma_1$

$$\frac{\frac{((C \rightarrow B) \rightarrow O) \rightarrow O}{O}}{((C \rightarrow B) \rightarrow O) \rightarrow O} \quad \Sigma_1$$

then let  $\Sigma^* = [C]$

$$\frac{C \rightarrow B}{B} \quad \Sigma$$

Case 3: If  $\Sigma = [C]$

$$\frac{\frac{\frac{(A \rightarrow O) \rightarrow O}{O}}{(C \rightarrow A) \rightarrow O}}{(C \rightarrow O) \rightarrow O} \quad \Sigma_1}{\frac{\frac{(A \rightarrow O) \rightarrow O}{O}}{(C \rightarrow A) \rightarrow O} \quad \Sigma_2} {[(C \rightarrow A)] [C]} \quad \Sigma_3$$

then let  $\Sigma^* = [A \rightarrow O]$



Case 5: If  $\Sigma = \frac{\Sigma_1}{\frac{\neg\neg A}{A}}$  then let  $\Sigma^* =$

$$\frac{\Sigma_1^*}{\frac{((\neg\neg A)^* \rightarrow \mathcal{O}) \rightarrow \mathcal{O}}{\mathcal{O}}} \quad \frac{\frac{\frac{\frac{[A^*]}{\mathcal{O}} \quad [A^* \rightarrow \mathcal{O}]}{(\perp \rightarrow \mathcal{O}) \rightarrow \mathcal{O}}}{A^* \rightarrow (\perp \rightarrow \mathcal{O}) \rightarrow \mathcal{O}}}{(\perp \rightarrow \mathcal{O}) \rightarrow \mathcal{O}}}{\frac{\mathcal{O}}{(\neg\neg A)^* \rightarrow \mathcal{O}}}$$

It is easy to check that in each case  $\widetilde{\Lambda}(\Sigma) = \widetilde{\Lambda}(\Sigma^*)$ .

## 5 Ongoing Research

The addition of an unrestricted  $\eta$  rule to the untyped  $\Lambda_v$ -calculus causes the theory to become inconsistent. It is conjectured that this is not the case when  $\eta$  is added to the reduction rules for C-derivations.

**Conjecture 1** *The reduction relation  $\Longrightarrow_c$  remains Church-Rosser with the addition of the  $\eta$  rule.*

If this conjecture is true, then the following theorem can be proved by a straightforward, but tedious, case analysis.

**Theorem 4** *If  $\Sigma_1$  and  $\Sigma_2$  are closed C-derivations with  $\Sigma_1 \Longrightarrow_c \Sigma_2$ , then  $\Sigma_1^* =_I \Sigma_2^*$ .*

Prawitz [Pra65] proves a normal form theorem for classical proofs where the reduction rule pushes instances of the  $\perp_c$  rule *further* from the root of a proof, rather than *closer* to the root, as is the case with the  $\Longrightarrow_c$  rules. A  $\Lambda$ -C analogue to Prawitz's rule could be written as  $\mathcal{C}(\lambda x.M) \rightarrow \lambda n.\mathcal{C}(\lambda k.(\lambda x.M)(\lambda f.k(fn)))$ . We conjecture that the reduction relation  $\Longrightarrow_c$  is strongly normalizing.

**Conjecture 2** *The reduction relation  $\Longrightarrow_c$  is strongly normalizing.*

Finally, applications of the correspondence between  $\Lambda$ -C-terms and classical proofs to program development are being investigated.

## References

- [BC85] Joseph L. Bates and Robert L. Constable. Proofs as programs. *ACM Trans. Prog. Lang and Sys.*, 7(1):113–136, 1985.
- [Con86] Robert L. Constable, et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.
- [FFKD86] M. Felleisen, D.P. Freidman, E. Kohlbecker, and B. Duba. Reasoning with continuations. In *First Symposium on Logic in Computer Science*, pages 131–141, IEEE, 1986.
- [FFKD87] M. Felleisen, D.P. Freidman, E. Kohlbecker, and B. Duba. A syntactic theory of sequential control. *Theoretical Computer Science*, 52(3):205–237, 1987.
- [Fis72] M. J. Fischer. Lambda calculus schemata. In *Proc. ACM Conference on Proving Assertions About Programs*, pages 104–109, 1972. SIGPLAN Notices 7.1.
- [How80] W. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus, and Formalism*, pages 479–490, Academic Press, NY, 1980.
- [Moh86] Christine Mohring. Algorithm development in the Calculus of Constructions. In *Proceedings of the First Symposium on Logic in Computer Science*, 1986.
- [MW85] A. R. Meyer and M. Wand. Continuation semantics in typed lambda-calculi (summary). In R. Parikh, editor, *Logics of Programs*, pages 219–224, Springer-Verlag, 1985. Lecture Notes in Computer Science, Volume 193.
- [Plo75] Gordon Plotkin. Call-by-name, call-by-value and the  $\lambda$ -calculus. *Theoretical Computer Science*, 1:125–159, 1975.
- [Pra65] Dag Prawitz. *Natural Deduction*. Almquist and Wiksell, 1965.
- [Ste72] Sören Stenlund. *Combinators, Lambda-Terms and Proof Theory*. D. Reidel, Dordrecht, Holland, 1972.

Rohit Parikh  
Program chair



Yours sincerely,

Thank you anyhow for submitting your paper to LICS 89 and I hope that you will attend the meeting in June.

was not among those selected by the program committee for presentation in June. Far more good papers were submitted than the program could possibly accommodate and we were forced to reject papers that were clearly publishable in a journal.

*In a Journal Correspondence between A-C-Terms + Chemical Terms*

I am sorry to have to tell you that your paper

Dear Author:

December 20, 1988

Department of Computer and Information Science

