Reasoning about Programs in Continuation-Passing Style

Amr Sabry   Matthias Felleisen
Department of Computer Science
Rice University
Houston, TX 77251-1892

Technical Report (*Rice TR 92-180*).

A preliminary version of this paper appeared in
Lisp and Functional Programming
June 22-24, 1992, San Francisco, California.

First Edition: May 28, 1992

Reasoning about Programs in Continuation-Passing Style

Amr Sabry   Matthias Felleisen *

Department of Computer Science
Rice University
Houston, TX 77251-1892

Abstract

Plotkin's λ-value calculus is sound but incomplete for reasoning about $\beta\eta$-transformations on programs in continuation-passing style (CPS). To find a complete extension, we define a new, compactifying CPS transformation and an "inverse" mapping, *un*-CPS, both of which are interesting in their own right. Using the new CPS transformation, we can determine the precise language of CPS terms closed under $\beta\eta$-transformations. Using the *un*-CPS transformation, we can derive a set of axioms such that every equation between source programs is provable if and only if $\beta\eta$ can prove the corresponding equation between CPS programs. The extended calculus is equivalent to an untyped variant of Moggi's computational λ-calculus.

1 Compiling with CPS

Many modern compilers for higher-order functional languages [1, 16, 24, 25] utilize some variant of the Fischer-Reynolds continuation-passing style (CPS) transformation [10, 23]. Once a program is in continuation-passing style, these compilers usually perform code optimizations via local transformations. Typical examples of such optimizations are loop unrolling, procedure inlining, and partial evaluation.

In the terminology of the λ-calculus, optimizations generally correspond to (sequences of) β- and η-reductions. Hence, a natural question to ask is whether reductions on CPS programs correspond to known transformations of source programs. If so, optimizations of CPS programs could be understood and reported in terms of the original program as opposed to its rather complicated CPS version. In particular, compilers that do *not* use the CPS transformation, e.g., Chez Scheme [15] or Zinc [18], could benefit by implementing transformations of source programs that correspond to transformations of CPS programs.

Technically speaking, we are addressing the following question: which calculus can prove $M = N$ for by-value expressions M and N, if $cps(M) = cps(N)$ is provable in the (by-value or by-name) λ-calculus? As Plotkin [22] showed in 1974, the λ_v-calculus does not suffice. Thus we refine this question as follows:

*Both authors were supported in part by NSF grant CCR 89-17022 and by Texas ATP grant 91-003604014.

Is there a set of axioms, A, that extends the call-by-value λ-calculus such that:

$$\boldsymbol{\lambda}_v A \vdash M = N \textit{ iff } \boldsymbol{\lambda}\beta\eta \vdash cps(M) = cps(N).$$

Such a correspondence theorem would be similar to the correspondence theorems for the λ-calculus and combinatory logic [2: ch 7], and the λ_v-calculus and by-value combinatory logic [13]. In analogy to model theory, we call the left-to-right direction *soundness* and the right-to-left direction *completeness* since the CPS transformation is often taken as the definition of a call-by-value semantics.

To derive A, we proceed in three steps:

1. First, we develop a CPS transformation that produces a canonical form of CPS programs. The new transformation also produces the smallest possible output of all known CPS transformations, without reducing any of the original (source) redexes.

2. Second, we develop an *un*-CPS transformation that maps canonical CPS programs and their derivations back to the original language. As Danvy and Lawall [3, 6] convincingly argue, this translation from CPS to *direct* terms is useful in its own right.

3. Finally, by studying the connection between the CPS and *un*-CPS transformations, we systematically derive A. The extended λ-value calculus is equivalent to an untyped variant of Moggi's [21] computational λ-calculus.

The next section introduces the basic terminology and notation of the λ-calculus and its semantics. The third section is a short history of CPS transformations. In Section 4, we formalize the problem and outline our approach to the solution. Section 5 is about the CPS language and its properties. It also contains the definition of our new CPS transformation. Section 6 presents the "inverse" mapping and its exact relation to the CPS transformation. Our main results, the extension of the λ_v-calculus and its completeness, are the subject of Section 7. The last two sections present corollaries of our result. The first discusses the typed and untyped versions of the calculi and their soundness with respect to observational equivalence. The second is a correspondence between the extension of the λ_v-calculus and the CPS type structure.

2 Λ: Calculi and Semantics

The language of the *pure* lambda calculus, Λ [2], consists of variables, λ-abstractions, and applications. The set of terms, M, is generated inductively over an infinite set of variables, *Vars*:

$$\begin{array}{rcll} M & ::= & V \mid (M\ M) & (\Lambda) \\ V & ::= & x \mid (\lambda x.M) & (\mathit{Values}) \\ x & \in & \mathit{Vars} & \end{array}$$

A term is either a value, V, or an application. Values consist of variables, drawn from the set *Vars*, and λ-abstractions.

We adopt Barendregt's [2: ch 2, 3] notation and terminology for this syntax. Thus, in the abstraction $(\lambda x.M)$, the variable x is *bound* in M. Variables that are not bound by a λ-abstraction are *free*; the set of free variables in a term M is $FV(M)$. A term is *closed* if it has no free variables; the set of closed terms is denoted by Λ^0. We identify terms modulo bound variables, and we assume that free and bound variables do not interfere in definitions or theorems. In short, we follow common practice and work with the quotient of Λ under α-equivalence. We write $M \equiv N$ for α-equivalent terms M and N.

The expression $M[x := N]$ is the result of the capture-free substitution of all free occurrences of x in M by N. For example, $(\lambda x.xz)[z := (\lambda y.x)] \equiv (\lambda u.u(\lambda y.x))$. A *context*, C, is a term with a "hole", $[\,]$, in the place of one subexpression. The operation of *filling* the context C with an expression M yields the term $C[M]$, possibly capturing some free variables of M in the process. Thus, the result of filling $(\lambda x.x[\,])$ with $(\lambda y.x)$ is $(\lambda x.x(\lambda y.x))$.

Calculi: A λ-calculus is an equational theory over Λ with a finite number of axiom schemas and inference rules. The most familiar axiom schemas are the following notions of reductions:

$$\begin{array}{rcllr}
((\lambda x.M)\; N) & \longrightarrow & M[x := N] & N : \text{arbitrary} & (\beta) \\
((\lambda x.M)\; V) & \longrightarrow & M[x := V] & V : \text{Value} & (\beta_v) \\
\lambda x.Mx & \longrightarrow & M & x \notin FV(M) & (\eta) \\
\lambda x.Vx & \longrightarrow & V & x \notin FV(V) & (\eta_v)
\end{array}$$

The set of inference rules is identical for all λ-calculi. It extends the notions of reductions to an equivalence relation compatible with syntactic contexts:

$$\begin{array}{rcllr}
M \longrightarrow N & \Rightarrow & C[M] = C[N] & \text{for all contexts } C & (\textit{Compatibility}) \\
 & & M = M & & (\textit{Reflexivity}) \\
M = L, L = N & \Rightarrow & M = N & & (\textit{Transitivity}) \\
M = N & \Rightarrow & N = M & & (\textit{Symmetry})
\end{array}$$

The underlying set of axioms completely identifies a theory. For example, β generates the theory $\boldsymbol{\lambda}$, β_v generates the theory $\boldsymbol{\lambda}_v$, and the union of β and η generates the theory $\boldsymbol{\lambda}\beta\eta$. In general, we write $\boldsymbol{\lambda}A$ to refer to the theory generated by a set of axioms A. When a theory $\boldsymbol{\lambda}A$ proves an equation $M = N$, we write $\boldsymbol{\lambda}A \vdash M = N$. If the proof does not involve the inference rule *Symmetry*, we write $\boldsymbol{\lambda}A \vdash M \twoheadrightarrow N$.

A notion of reduction R is Church-Rosser (CR) if $\boldsymbol{\lambda}R \vdash M = N$ implies that there exists a term L such that both M and N reduce to L, i.e., $\boldsymbol{\lambda}R \vdash M \twoheadrightarrow L$ and $\boldsymbol{\lambda}R \vdash N \twoheadrightarrow L$. A term M is in R-normal form if there are no R-reductions starting with M.

Semantics: The semantics of the language Λ is a function, *eval*, from programs to answers. A *program* is a term with no free variables and, in practical languages, an *answer* is a member of the syntactic category of values. Typically, *eval* is defined via an abstract machine that manipulates abstract counterparts to machine stacks, stores, registers, etc. Examples are the SECD machine [17] and the CEK machine [7].

An equivalent method for specifying the semantics is based on the Curry-Feys Standard Reduction theorem [7, 22]. The Standard Reduction theorem defines a partial function,

$\longmapsto$, from programs to programs that corresponds to a single evaluation step of an abstract machine for Λ.

A standard step (i) decomposes the program into a context E and a leftmost-outermost redex R (not inside an abstraction), and (ii) fills E with the contractum of R. The special contexts, E, are *evaluation contexts* and have the following definition for the call-by-value and call-by-name variants of Λ respectively [7]:

$$\begin{aligned} E_v &::= \quad [\,] \mid (V\ E_v) \mid (E_v\ M) \\ E_n &::= \quad [\,] \mid (E_n\ M) \end{aligned}$$

Conceptually, the hole of an evaluation context, $[\,]$, points to the current instruction, which must be a β_v or β redex. The decomposition of M into $E[(V\ N)]$ where $(V\ N)$ is a redex means that the current instruction is $(V\ N)$ and that the rest of the computation (the continuation [7]!) is E. Since, a call-by-name language never evaluates arguments, evaluation contexts do not include contexts of the shape $(V\ E_n)$.

Given evaluation contexts, the definitions of the standard reduction functions for call-by-value and call-by-name respectively are as follows:

$$\begin{aligned} E_v[((\lambda x.M)\ V)] &\longmapsto_v E_v[M[x := V]] \\ E_n[((\lambda x.M)\ N)] &\longmapsto_n E_n[M[x := N]] \end{aligned}$$

A complete evaluation applies the single-step functions repeatedly and either reaches an answer or diverges. The notation $\longmapsto^*$ denotes the reflexive, transitive closure of the function $\longmapsto$. The semantics of Λ is defined as follows:

$$\begin{aligned} eval_v(M) &= V \text{ iff } M \longmapsto_v^* V && \text{(call-by-value)} \\ eval_n(M) &= V \text{ iff } M \longmapsto_n^* V && \text{(call-by-name)} \end{aligned}$$

For the definition of the semantics, η and η_v do not play any role. Their relevance for calculi is clarified in the next paragraph.

An important fact for the discussions below is that the syntax of the call-by-value language Λ can be redefined as follows:

$$\begin{aligned} M &::= V \mid E[(V\ V)] && (\Lambda) \\ V &::= x \mid (\lambda x.M) && (\mathit{Values}) \\ E &::= [\,] \mid (V\ E) \mid (E\ M) && (\mathit{EvCont}) \end{aligned}$$

The set of evaluation contexts has also the following equivalent definition:

$$E ::= [\,] \mid E[(V\ [\,])] \mid E[([\,]\ M)] \qquad (\mathit{EvCont})$$

We use all the definitions interchangeably. Moreover, we extend the notions of reductions to evaluation contexts by treating the hole as a placeholder for an arbitrary expression. For example, the reduction:

$$((\lambda x.(x\ y))\ [\,]) \longrightarrow ([\,]\ y)$$

is a β-reduction; it is *not* a β_v-reduction.

Observational Equivalence: Not only do calculi define the semantics of Λ, but they are also useful for proving the correctness of some *optimizations*. Abstractly, an optimization of a program $C[M]$ is the replacement of M by a "more efficient" expression N such that a programmer cannot distinguish the observational behavior of the programs $C[M]$ and $C[N]$. The observational behavior of a program includes its termination behavior and its value when it terminates; it does not include execution speed. Formally, two expressions M and N are observationally equivalent, $M \cong_x N$ (for $x = v$ or $x = n$), if the following condition holds:

> For all contexts C such that both $C[M]$ and $C[N]$ are programs, either both $eval_x(C[M])$ and $eval_x(C[N])$ are defined or both are undefined.

It is undecidable to determine whether two expressions are observationally equivalent. However, $\boldsymbol{\lambda}_v$ and $\boldsymbol{\lambda}$ are two typical (weak) examples of theories that are sound with respect to observational equivalence.

Theorem 2.1 (Plotkin) *Let $M, N \in \Lambda$.*

1. *If $\boldsymbol{\lambda}_v \vdash M = N$ then $M \cong_v N$.*
2. *If $\boldsymbol{\lambda} \vdash M = N$ then $M \cong_n N$.*

The soundness of extensions of $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}_v$ with η and η_v, respectively, depends on the circumstances. The axiom η_v is sound with respect to call-by-value observational equivalence for Λ. If we extend Λ with constants, η_v may be unsound. For an example, consider a dynamically typed language with numerals and a predicate *integer?*. The latter can distinguish 3 and $(\lambda x.3\ x)$, yet, the η_v axiom identifies the two terms. In a typed setting, η_v is generally sound, independent of the parameter-passing technique.

The axiom η, on the other hand, fails to be sound with respect to call-by-name observational equivalence even in a pure language. For example, if Ω is a diverging term, then $(\lambda x.\Omega x)$ reduces to Ω but the two are clearly observationally distinct terms. Indeed, η is only sound in a typed language that does not permit the observation of the termination behavior of higher-type expressions.

3 The Origins and Practice of CPS

The idea of transforming programs to "continuation-passing style" appeared in the mid-sixties. For a few years, the transformation remained part of the folklore of computer science until Fischer and Reynolds codified it in 1972.

Fischer [10] studied two implementation strategies for Λ: a heap-based retention strategy, in which all variable bindings are retained until no longer needed, and a stack-based deletion strategy, in which variable bindings are destroyed when control leaves the procedure (or block) in which they were created. He concluded that

> no real power is lost in restricting oneself to a deletion strategy implementation, for any program can be translated into an equivalent one which will work correctly under such an implementation [10: 104].

The translation is defined as follows.

Definition 3.1. (*Fischer CPS*) Let k, m, $n \in \mathit{Vars}$ be variables that do not occur in the argument to $\mathcal{F}$.

$$\begin{array}{rcl}\mathcal{F} : \Lambda & \rightarrow & \Lambda \\ \mathcal{F}[\![V]\!] & = & \lambda k.k\ \Psi[\![V]\!] \\ \mathcal{F}[\![MN]\!] & = & \lambda k.\mathcal{F}[\![M]\!]\ (\lambda m.\mathcal{F}[\![N]\!]\ \lambda n.(m\ k)\ n) \\ \\ \Psi[\![x]\!] & = & x \\ \Psi[\![\lambda x.M]\!] & = & \lambda k.\lambda x.\mathcal{F}[\![M]\!]\ k\end{array}$$

∎

Reynolds [23] investigated definitional interpreters for higher-order languages. One of his goals was the desire to liberate the definition of a language from the parameter-passing technique of the defining language. He developed a constructive (but informal) method to transform an interpreter such that it becomes indifferent to whether the underlying parameter passing technique is call-by-value or call-by-name. His transformation is essentially the same transformation as Fischer's $\mathcal{F}$.[1] Plotkin [22] later proved Reynolds' ideas correct.

Theorem 3.2 (Plotkin [22]) *Let* $M \in \Lambda$.

Simulation: $\Psi[\![eval_v(M)]\!] = eval_n(\mathcal{F}[\![M]\!]\ (\lambda x.x))$

Indifference: $eval_n(\mathcal{F}[\![M]\!]\ (\lambda x.x)) = eval_v(\mathcal{F}[\![M]\!]\ (\lambda x.x))$

The Simulation theorem shows that the evaluation of the CPS program produces correct outputs. The Indifference theorem establishes that this evaluation yields the same result under call-by-value and call-by-name.

The main disadvantage of the Fischer CPS transformation is the excessive number of redexes it introduces in the output. For example,

$$\begin{array}{rcl}\mathcal{F}[\![((\lambda x.x)\ (y\ y))]\!] & = & \lambda k.((\lambda k.k\ \lambda k.\lambda x.((\lambda k.kx)\ k)) \\ & & \quad (\lambda m.((\lambda k.((\lambda k.ky)\ \lambda m.((\lambda k.ky)\ \lambda n.(m\ k)\ n))) \\ & & \qquad (\lambda n.(m\ k)\ n)))).\end{array}$$

Although the original term contains one λ-abstraction and no β_v-redexes, its CPS counterpart contains a large number of both. Plotkin [22] referred to the new redexes as *administrative* redexes because an evaluator must always reduce them before re-establishing β_v-redexes that were present in the source term.

From both a theoretical and a practical perspective, the presence of the administrative redexes is undesirable. On the theoretical side, they complicate reasoning about CPS programs. For example, Plotkin [22] finds it necessary to define an improved CPS transformation exclusively for the proof of Theorem 3.2 above. On the practical side, code generation phases in compilers favor smaller, i.e., more manageable, programs. Hence,

[1] In Reynolds's transformation the continuation is the *second* argument to a procedure.

"practical" CPS transformations [1, 5, 16, 24, 25] use special algorithms to minimize the size of their outputs.

In essence, all practical CPS transformations are conceptually equivalent to the following two-pass CPS transformation:[2]

- First, "mark" the new λ-abstractions in the output of the Fischer CPS to identify administrative redexes, and then
- reduce all administrative redexes.

Source redexes should remain intact because unrestricted reductions could cause non-termination. The remainder of this section codifies these ideas in a simple manner.

Formally, the first pass of the two-pass CPS is the following modified Fischer CPS transformation.

Definition 3.3. (*Modified Fischer CPS*) Let $k, m, n \in \mathit{Vars}$ be as in Definition 3.1.

$$\begin{array}{rcl} \mathcal{F}[\![V]\!] & = & \overline{\lambda} k.k\ \Psi[\![V]\!] \\ \mathcal{F}[\![MN]\!] & = & \overline{\lambda} k.\mathcal{F}[\![M]\!]\ (\overline{\lambda} m.\mathcal{F}[\![N]\!]\ \overline{\lambda} n.(m\ k)\ n) \\ \\ \Psi[\![x]\!] & = & x \\ \Psi[\![\lambda x.M]\!] & = & \overline{\lambda} k.\lambda x.\mathcal{F}[\![M]\!]\ k \end{array}$$

∎

An overline decorates λ-abstractions that were not present in the original term. An administrative reduction is simply one that involves decorated abstractions:

$$\begin{array}{rclll} ((\overline{\lambda} x.M)\ N) & \longrightarrow & M[x := N] & & (\overline{\beta}) \\ (\overline{\lambda} x.Mx) & \longrightarrow & M & x \notin FV(M) & (\overline{\eta}) \end{array}$$

The complete definition of the two-pass CPS transformation, $\mathcal{F}2$, is the following.

Definition 3.4. (*Two-Pass CPS*) $\mathcal{F}2[\![M]\!] = P$ iff $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![M]\!] = P$ where P is in $\overline{\beta}\overline{\eta}$-normal form. ∎

The following proposition establishes that $\mathcal{F}2$ is well-defined.

Proposition 3.5 *$\mathcal{F}2$ is a total function.*

Proof. By Lemma 3.6, $\overline{\beta}\overline{\eta}$-normal forms are unique. Therefore, the relation $\mathcal{F}2$ is a function. Moreover, by Lemma 3.7, all reduction paths starting at $\mathcal{F}[\![M]\!]$ for $M \in \Lambda$ terminate. Hence, $\mathcal{F}2$ is a total function. ∎

[2] See also the discussion of this idea by Danvy and Filinski [5].

Lemma 3.6 *Let P and Q be in $\overline{\beta}\overline{\eta}$-normal form. If $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![M]\!] \longrightarrow\!\!\!\!\rightarrow P$ and $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![M]\!] \longrightarrow\!\!\!\!\rightarrow Q$, then $P \equiv Q$.*

Proof. The proof is a consequence of the Church-Rosser theorem for $\beta\eta$ [2]. ∎

It remains to establish that all $\overline{\beta}\overline{\eta}$-reduction paths terminate.

Lemma 3.7 *Let $M \in \Lambda$. If $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![M]\!] \equiv M_0 \longrightarrow M_1 \longrightarrow M_2 \cdots$ then:*

1. *for all M_i, the bound variable of a $\overline{\lambda}$-abstraction occurs exactly once in the body,*
2. *for all $i \geq 0$, M_{i+1} has one less $\overline{\lambda}$-abstraction than M_i, and*
3. *for some finite n, M_n is in $\overline{\beta}\overline{\eta}$-normal form.*

Proof Sketch. The first claim is initially true by construction, and is preserved by $\overline{\beta}\overline{\eta}$-reductions. It implies that reductions cannot eliminate or duplicate subterms. Therefore, the second claim holds. The last claim follows by induction on the number of $\overline{\lambda}$-abstractions in $\mathcal{F}[\![M]\!]$. ∎

The output of $\mathcal{F}2$ is extremely compact. For example, applying $\mathcal{F}2$ to $(((\lambda x.\lambda y.x)\ a)\ b)$ yields the term:

$$M \stackrel{df}{=} \lambda k.((\lambda x.((\lambda y.kx)\ b))\ a)$$

For the same example, both Steele's Rabbit transformation [25] and the Danvy/Filinski transformation [5] yield the term:[3]

$$N \stackrel{df}{=} \lambda k.((\lambda k_1 x.(k_1\ \lambda k_2 y.k_2 x))\ (\lambda m.mkb)\ a).$$

The evaluation of M requires two β-reductions:

$$M \longrightarrow \lambda k.((\lambda y.ka)\ b) \longrightarrow \lambda k.ka,$$

while the evaluation of N requires three (binary) β-reductions:

$$\begin{aligned} N &\longrightarrow \lambda k.((\lambda m.mkb)\ (\lambda k_2 y.k_2 a)) \\ &\longrightarrow \lambda k.((\lambda k_2 y.k_2 a)\ k\ b) \\ &\longrightarrow \lambda k.ka. \end{aligned}$$

Since the extra (administrative) reduction in the evaluation of N is completely predictable from the source term, the function $\mathcal{F}2$ optimizes it away.

[3] This is slightly inaccurate. In both Steele's Rabbit and the Danvy/Filinski CPS transformations, the continuation is the second parameter to a procedure. Thus, their output is actually:

$$\lambda k.((\lambda x k_1.(k_1\ \lambda y k_2.k_2 x))\ a\ (\lambda m.mbk)).$$

Even though this term only contains source redexes, we could still optimize it by equational reasoning as the following derivation shows:

$$\begin{aligned} & \lambda k.((\lambda x k_1.(k_1 \lambda y k_2.k_2 x))\ a\ (\lambda m.mbk)) \\ \longrightarrow\ & \lambda k.((\lambda m.mbk)\ (\lambda y k_2.k_2 a)) \\ \longrightarrow\ & \lambda k.((\lambda y k_2.k_2 a)\ b\ k) \\ \longrightarrow\ & \lambda k.ka \\ \longleftarrow\ & \lambda k.((\lambda y.ka)\ b) \\ \longleftarrow\ & \lambda k.((\lambda x.((\lambda y.kx)\ b))\ a). \end{aligned}$$

Indeed, the "net" effect of such transformations is that of performing administrative reductions only.

4 Transforming CPS programs

With the elimination of all administrative redexes, we can turn our attention to "interesting" $\beta\eta$-transformations on CPS programs.

Plotkin [22] was the first to offer some insights about the relation between reductions on source terms and CPS terms. In his comparative study of equational theories for call-by-value languages and call-by-name languages, he proved the following theorem.

Theorem 4.1 (Plotkin [22]) *Let $M, N \in \Lambda$.*

1. $\boldsymbol{\lambda}_v \vdash M = N$ *implies* $\boldsymbol{\lambda}_v \vdash \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$;
2. $\boldsymbol{\lambda}_v \vdash \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$ *does not imply* $\boldsymbol{\lambda}_v \vdash M = N$;
3. $\boldsymbol{\lambda}_v \vdash \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$ *iff* $\boldsymbol{\lambda} \vdash \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$.

In short, β-reductions prove more equations on CPS terms than β_v-reductions prove on source terms. The effect of η-reductions is unknown. Moreover, the induced observational equivalence relations do not correspond.

Corollary 4.2 (Plotkin [22]) *Let $M, N \in \Lambda$.*

1. $\mathcal{F}[\![M]\!] \cong_n \mathcal{F}[\![N]\!]$ *implies* $M \cong_v N$;
2. $M \cong_v N$ *does not imply* $\mathcal{F}[\![M]\!] \cong_n \mathcal{F}[\![N]\!]$.

In the second part of the corollary, the terms $\mathcal{F}[\![M]\!]$ and $\mathcal{F}[\![N]\!]$ fail to be observationally equivalent in contexts that manipulate the continuation in "non-standard" ways. We return to the observational equivalence relations and their correspondence in Section 8.

Our more immediate goal is to derive a set of axioms A such that:

$$\boldsymbol{\lambda}_v A \vdash M = N \quad \text{iff} \quad \boldsymbol{\lambda}\beta\eta \vdash \mathcal{F}2[\![M]\!] = \mathcal{F}2[\![N]\!].$$

We illustrate some of the complications that this problem poses with a specific reduction on CPS terms:

$$\lambda k.((\lambda x.((x\ k)\ z))\ (\lambda k.\lambda y.ky)) \longrightarrow \lambda k.(((\lambda k.\lambda y.ky)\ k)\ z).$$

By inspection, the left-hand side is:

$$\mathcal{F}2[\![((\lambda x.xz)\ (\lambda y.y))]\!].$$

A quick glance at the right hand side reveals that it contains an administrative redex and hence cannot be $\mathcal{F}2[\![M]\!]$ for any $M \in \Lambda$. The right hand side is, however, provably equal to a number of CPS terms:

$$\boldsymbol{\lambda}\beta\eta \vdash \mathcal{F}2[\![((\lambda a.((\lambda x.xa)\ (\lambda y.y)))\ z)]\!] = \mathcal{F}2[\![((\lambda y.y)\ z)]\!] = \lambda k.(((\lambda k.\lambda y.ky)\ k)\ z).$$

Assuming we choose $((\lambda y.y)\ z)$ as the "inverse" of the right hand side, then the CPS reduction corresponds to the following β_v-reduction:

$$((\lambda x.xz)\ (\lambda y.y)) \longrightarrow ((\lambda y.y)\ z)$$

on source terms. The other choice corresponds to a β_v-*expansion*, which is clearly undesirable.

Inspired by the above example, we proceed as follows:

1. We explicitly define the set of CPS terms. The definition relies on a one-pass CPS transformation equivalent to $\mathcal{F}2$ (Section 5).

2. We define an "inverse" CPS transformation and formalize its precise relationship to the CPS transformation (Section 6).

3. We derive the set A. For each notion of reduction $P \longrightarrow Q$ on CPS terms, we apply the inverse transformation to P and Q and get the source terms M and N. If $\boldsymbol{\lambda}_v \vdash M = N$, then we are done. Otherwise, we add appropriate reductions to A (Section 7).

5 The CPS language

The one-pass CPS transformation should combine the modified Fischer transformation with the application of $\overline{\beta}$- and $\overline{\eta}$-reductions. An informal description of what these reductions accomplish will clarify the nature of such a function.

The most informative kind of administrative redexes appears in the translation of a redex $((\lambda x.M)\ V)$ in an arbitrary continuation K:

$$\begin{array}{l}((\overline{\lambda}k.((\overline{\lambda}k.k\ (\overline{\lambda}k.\lambda x.\mathcal{F}[\![M]\!]\ k))\\ \qquad (\overline{\lambda}m.((\overline{\lambda}k.k\ \Psi[\![V]\!])\ (\overline{\lambda}n.(mk)n)))))\\ \ K).\end{array}$$

The expression reduces to:

$$(((\overline{\lambda}k.\lambda x.\mathcal{F}[\![M]\!]\ k)\ K)\ \Psi[\![V]\!])$$

via a number of $\overline{\beta}\overline{\eta}$-reductions. At this point, the following $\overline{\beta}$-reduction takes place:

$$(((\overline{\lambda}k.\lambda x.\mathcal{F}[\![M]\!]\ k)\ K)\ \Psi[\![V]\!]) \longrightarrow ((\lambda x.\mathcal{F}[\![M]\!]\ K)\ \Psi[\![V]\!]),$$

i.e., the image of the abstraction absorbs the continuation of the application. For the source terms, this means that the *body* of an abstraction in application position absorbs the syntactic representation of the continuation, which is the evaluation context of the redex [7]. Thus, a program of the shape $E[((\lambda x.M)\ V)]$ where E represents K, must be translated as if it had been written as $((\lambda x.E[M])\ V)$.

Put differently, our CPS transformation "symbolically" evaluates redexes by lifting them to the root of the program. For applications of values to values inside of λ-abstractions, this means of course that it takes the evaluation contexts with respect to the closest λ, which will become the root of the program once the redex is discharged. The resulting transformation, $\mathcal{C}_k$, is parametrized over a variable k that represents the current continuation.

Definition 5.1. $(\mathcal{C}_k, \Phi, \mathcal{K}_k)$ The CPS transformation uses three mutually recursive functions: $\mathcal{C}_k$ to transform terms, Φ to transform values, and $\mathcal{K}_k$ to transform evaluation contexts. Let $k, u_i \in \mathit{Vars}$ be variables that do not occur in the argument to $\mathcal{C}_k$.[4]

$$
\begin{array}{rcl}
\mathcal{C}_k : \Lambda & \rightarrow & \Lambda \\
\mathcal{C}_k[\![V]\!] & = & (k\ \Phi[\![V]\!]) \\
\mathcal{C}_k[\![E[(x\ V)]]\!] & = & ((x\ \mathcal{K}_k[\![E]\!])\ \Phi[\![V]\!]) \\
\mathcal{C}_k[\![E[((\lambda x.M)\ V)]]\!] & = & ((\lambda x.\mathcal{C}_k[\![E[M]]\!])\ \Phi[\![V]\!])
\end{array}
$$

$$
\begin{array}{rcl}
\Phi[\![x]\!] & = & x \\
\Phi[\![\lambda x.M]\!] & = & \overline{\lambda} k.\lambda x.\mathcal{C}_k[\![M]\!]
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{K}_k[\![[\ ]]\!] & = & k \\
\mathcal{K}_k[\![E[(x\ [\ ])]]\!] & = & (x\ \mathcal{K}_k[\![E]\!]) \\
\mathcal{K}_k[\![E[((\lambda x.M)\ [\ ])]]\!] & = & (\lambda x.\mathcal{C}_k[\![E[M]]\!]) \\
\mathcal{K}_k[\![E[([\ ]\ M)]]\!] & = & (\overline{\lambda} u_i.\mathcal{C}_k[\![E[(u_i\ M)]]\!])
\end{array}
$$

∎

An informal examination of the above definition reveals that the function is not defined by structural recursion. Still, it is relatively easy to check that with an appropriate notion of "size", the translation of every expression refers to the translation of a smaller expression. In particular, the size of $E[(u_i\ M)]$ is smaller than the size of $E[([\ ]\ M)]$ because the empty context always replaces an application. The formal definition of "size" is as follows.

Definition 5.2. (*Size*) The *size* of a term M, $|M|$, is the number of variables in M (including binding occurrences). The *size* of a context E, $|E|$, is the number of variables in E (including binding occurrences) plus 2. ∎

As expected, the output of $\mathcal{C}_k$ is equivalent to the output of $\mathcal{F}2$, which also proves that $\mathcal{C}_k$ is well-defined.

Proposition 5.3 *Let $M \in \Lambda$. Then, $\mathcal{F}2[\![M]\!] \equiv \lambda k.\mathcal{C}_k[\![M]\!]$.*

Proof. The essential steps in the proof are:

1. Define a natural extension of the modified Fischer transformation that accepts evaluation contexts: (Definition 5.4). Next, prove that:

$$\boldsymbol{\lambda}\overline{\beta}\ \vdash\ (\mathcal{F}[\![E[M]]\!]\ k) = (\mathcal{F}[\![M]\!]\ (\mathcal{F}[\![E]\!]\ k)) \qquad \text{(Lemma 5.5).}$$

2. Using the previous result, prove that:

$$\boldsymbol{\lambda}\overline{\beta}\overline{\eta}\ \vdash\ (\mathcal{F}[\![M]\!]\ k) = \mathcal{C}_k[\![M]\!] \qquad \text{(Lemma 5.6).}$$

[4]The CPS transformation $\mathcal{C}_k$ is related to the CPS transformation by Friedman, Wand, and Haynes [11: ch 8], but differs significantly in several technical aspects.

3. The result follows because $\mathcal{C}_k[\![M]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form (Lemma 5.6) and $\overline{\beta}\overline{\eta}$-normal forms are unique (Lemma 3.6). ∎

The extension of $\mathcal{F}$ that accepts evaluation contexts is as follows.

Definition 5.4. (*Extension of the Fischer CPS*) Let $k, m, n \in \mathit{Vars}$ as in Definition 3.1:

$$\begin{aligned}
\mathcal{F} : \mathit{EvCont} &\rightarrow \Lambda \\
\mathcal{F}[\![[\,]]\!] &= \overline{\lambda} k.k \\
\mathcal{F}[\![(V\ E)]\!] &= \overline{\lambda} k.\mathcal{F}[\![V]\!]\ \overline{\lambda} m.\mathcal{F}[\![E]\!]\ \overline{\lambda} n.(m\ k)\ n \\
\mathcal{F}[\![(E\ M)]\!] &= \overline{\lambda} k.\mathcal{F}[\![E]\!]\ \overline{\lambda} m.\mathcal{F}[\![M]\!]\ \overline{\lambda} n.(m\ k)\ n
\end{aligned}$$

∎

The following lemma establishes an important property of evaluation contexts. Intuitively, the evaluation of $E[M]$ in a continuation K is identical to the evaluation of M in a continuation that combines E and K.

Lemma 5.5 *Let $M \in \Lambda$, $E \in \mathit{EvCont}$. Then,* $\boldsymbol{\lambda}\overline{\beta} \vdash (\mathcal{F}[\![E[M]]\!]\ k) = (\mathcal{F}[\![M]\!]\ (\mathcal{F}[\![E]\!]\ k))$.

Proof Idea. The proof is by induction on the structure of E. ∎

It remains to establish that the output of $\mathcal{F}$ is provably equal to the output of $\mathcal{C}_k$ and that the latter is in $\overline{\beta}\overline{\eta}$-normal form.

Lemma 5.6 *Let $M \in \Lambda$, $E \in \mathit{EvCont}$. Then,*

1. $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![M]\!]\ k) = \mathcal{C}_k[\![M]\!]$ *and* $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![E]\!]\ k) = \mathcal{K}_k[\![E]\!]$;
2. $\mathcal{C}_k[\![M]\!]$ *and* $\mathcal{K}_k[\![E]\!]$ *are in $\overline{\beta}\overline{\eta}$-normal form.*

Proof. The proof is by induction on the size of G, where G is the argument to $\mathcal{C}_k$ or $\mathcal{K}_k$. We proceed by case analysis on the possible inputs:

1. $G \equiv V$: then, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = ((\overline{\lambda} k.k\ \Psi[\![G]\!])\ k) = (k\ \Psi[\![G]\!])$. By cases:
   (a) $G \equiv x$: then $(k\ \Psi[\![G]\!]) \equiv (k\ x) \equiv \mathcal{C}_k[\![x]\!]$. Moreover, $\mathcal{C}_k[\![x]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form.
   (b) $G \equiv \lambda x.M$: then $(k\ \Psi[\![G]\!]) \equiv (k\ \overline{\lambda} c.\lambda x.\mathcal{F}[\![M]\!] c)$. By the inductive hypothesis $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![M]\!]\ c) = \mathcal{C}_c[\![M]\!]$ and $\mathcal{C}_c[\![M]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form. Therefore $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![\lambda x.M]\!]\ k) = \mathcal{C}_k[\![\lambda x.M]\!]$ and $\mathcal{C}_k[\![\lambda x.M]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form.
2. $G \equiv E[(x\ V)]$: then $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = (\mathcal{F}[\![(x\ V)]\!]\ (\mathcal{F}[\![E]\!]\ k))$ by Lemma 5.5. The latter term is provably equal to $((x\ \mathcal{F}[\![E]\!] k)\ \Psi[\![V]\!])$. There are two cases:
   (a) $V \notin \mathit{Vars}$: then $|E| < |G|$. By the inductive hypothesis, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![E]\!] k = \mathcal{K}_k[\![E]\!]$ and $\mathcal{K}_k[\![E]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form. By an argument similar to case 1, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \Psi[\![V]\!] = \Phi[\![V]\!]$ and $\Phi[\![V]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form. Both results follow since $\mathcal{C}_k[\![G]\!] \equiv ((x\ \mathcal{K}_k[\![E]\!])\ \Phi[\![V]\!])$.

(b) $V \in \mathit{Vars}$: then $|E| = |G|$ and the inductive hypothesis does not apply. By inlining the arguments in cases 4 to 7, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![E]\!]\ k) = \mathcal{K}_k[\![E]\!]$ and $\mathcal{K}_k[\![E]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form. The result follows as in subcase (a).

3. $G \equiv E[((\lambda x.M)\ V)]$: then $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash \mathcal{F}[\![G]\!]k = (\mathcal{F}[\![((\lambda x.M)\ V)]\!]\ (\mathcal{F}[\![E]\!]k))$ by Lemma 5.5. The latter expression is provably equal to $((\lambda x.\mathcal{F}[\![E[M]]\!]k)\ \Psi[\![V]\!])$. The result follows by the inductive hypothesis and an argument similar to case 1.
4. $G \equiv [\ ]$: then $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = ((\overline{\lambda} k.k)\ k) = k = \mathcal{K}_k[\![G]\!]$. Moreover $\mathcal{K}_k[\![G]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form.
5. $G \equiv E[(x\ [\ ])]$: then, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = (\mathcal{F}[\![(x\ [\ ])]\!]\ (\mathcal{F}[\![E]\!]k))$ by Lemma 5.5. The latter expression is provably equal to $(x\ \mathcal{F}[\![E]\!]k)$. The result follows by the inductive hypothesis.
6. $G \equiv E[((\lambda x.M)\ [\ ])]$: then $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = (\lambda x.\mathcal{F}[\![E[M]]\!]k)$ and the result follows by the inductive hypothesis.
7. $G \equiv E[([\ ]\ M)]$: then $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = (\mathcal{F}[\![([\ ]\ M)]\!]\ (\mathcal{F}[\![E]\!]k))$ by Lemma 5.5. The latter expression is provably equal to $(\overline{\lambda} u.\mathcal{F}[\![M]\!]\ (u\ (\mathcal{F}[\![E]\!]k)))$. By another application of Lemma 5.5, we get $(\overline{\lambda} u.(\mathcal{F}[\![E[(u\ M)]]\!]\ k))$. By the inductive hypothesis, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![E[(u\ M)]]\!]\ k) = \mathcal{C}_k[\![E[(u\ M)]]\!]$ and $\mathcal{C}_k[\![E[(u\ M)]]\!]$ is in $\overline{\beta}\overline{\eta}$-normal form. Therefore, $\boldsymbol{\lambda}\overline{\beta}\overline{\eta} \vdash (\mathcal{F}[\![G]\!]\ k) = \mathcal{K}_k[\![G]\!]$. Moreover, by a simple case analysis, $\mathcal{C}_k[\![E[(u\ M)]]\!]$ is never of the form $(K\ u)$ for some term K. Therefore, no new $\overline{\eta}$-redex is created in $\overline{\lambda} u.\mathcal{C}_k[\![E[(u\ M)]]\!]$ and the term in $\overline{\beta}\overline{\eta}$-normal form. ∎

With the completion of the analysis of the new CPS algorithms, the decorating overlines become irrelevant. Therefore, in the remainder of the paper, we ignore the distinction between λ and $\overline{\lambda}$.

Besides its pragmatic significance, the new CPS transformation simplifies the definition of the set of CPS terms. Specifically, our universe of discourse consists only of the terms that contribute to the proofs of equations of the form:

$$\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] = \mathcal{C}_k[\![N]\!].$$

Since $\beta\eta$ is *CR* [2], it is sufficient to consider equations of the form:

$$\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow P.$$

Hence, the interesting set of CPS terms is:

$$S \stackrel{df}{=} \{P \mid \exists M \in \Lambda.\ \boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow P\}.$$

The definition of the function $\mathcal{C}_k$ provides some insight about an inductive characterization of the set of CPS terms. According to the right hand sides of the equations in Definition 5.1, all terms in the CPS language are an application of a continuation to a value. Values are either variables or abstractions (continuation transformers). Continuations are either variables, or the result of the application of a value to a continuation, or a regular lambda abstraction. Therefore, we claim that S is generated by the following grammar.

Definition 5.7. (*CPS grammar*) Let $x \in \mathit{Vars} \backslash \{k\}$:

$$\begin{array}{rcll} P & ::= & (K\ W) & (\mathit{cps}(\Lambda)) \\ W & ::= & x \mid (\lambda k.K) & (\mathit{cps}(\mathit{Values})) \\ K & ::= & k \mid (W\ K) \mid (\lambda x.P) & (\mathit{cps}(\mathit{EvCont})) \end{array}$$

∎

Note: The special status reserved for the variable k ensures that the continuation parameter occurs exactly once in each abstraction $\lambda k.K$. A *program* in CPS form is a closed term of the form $((\lambda k.P)\,(\lambda x.x))$ where k is the special continuation paramter. When working with the quotient of the language under α-equivalence, the special status of k disappears. **End Note**

The following theorem establishes the equivalence of the two definitions of the set of CPS terms.

Theorem 5.8 $S = \mathit{cps}(\Lambda)$.

Proof. The left to right inclusion is the subject of Lemma 5.9. Lemma 5.13 deals with the opposite direction. ∎

Lemma 5.9 $S \subseteq \mathit{cps}(\Lambda)$.

Proof. Let $P \in S$. From the definition of the set S, there exists an $M \in \Lambda$ such that $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow P$ in n steps where $n \geq 0$. By induction on n, we prove that $P \in \mathit{cps}(\Lambda)$.

- $n = 0$, then $P \equiv \mathcal{C}_k[\![M]\!]$. The result follows by Lemma 5.10.
- $n = i + 1$, then $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow Q \longrightarrow P$ for some Q. By the inductive hypothesis, $Q \in \mathit{cps}(\Lambda)$. Moreover, by Lemma 5.11, $\beta\eta$-reductions preserve the syntactic categories in the CPS language. Therefore, $P \in \mathit{cps}(\Lambda)$. ∎

Lemma 5.10 *Let $M \in \Lambda$, $E \in \mathit{EvCont}$. Then, $\mathcal{C}_k[\![M]\!] \in \mathit{cps}(\Lambda)$, $\mathcal{K}_k[\![E]\!] \in \mathit{cps}(\mathit{EvCont})$.*

Proof Idea. The proof is by induction on the size of the argument to $\mathcal{C}_k$ or $\mathcal{K}_k$. It follows the same strategy as the proof of Lemma 5.6. ∎

It remains to establish that $\beta\eta$-reductions preserve the syntactic categories in the grammar of Definition 5.7. By inspection of the CPS grammar, the possible β- and η-reductions on CPS terms are as follows:

$$\begin{array}{rcll} ((\lambda x.P)\ W) & \longrightarrow & P[x := W] & & (\beta_w) \\ ((\lambda k.K_1)\ K_2) & \longrightarrow & K_1[k := K_2] & & (\beta_k) \\ (\lambda k.Wk) & \longrightarrow & W & & (\eta_w) \\ (\lambda x.Kx) & \longrightarrow & K & x \notin FV(K) & (\eta_k) \end{array}$$

Therefore, for $\mathit{cps}(\Lambda)$, $\beta = \beta_w \cup \beta_k$ and $\eta = \eta_w \cup \eta_k$.

Lemma 5.11 *Let $P_1 \in cps(\Lambda)$, $W_1 \in cps(Values)$, $K_1 \in cps(EvCont)$. Then,*

1. $\boldsymbol{\lambda}\beta\eta \vdash P_1 \longrightarrow P_2$ *implies* $P_2 \in cps(\Lambda)$.
2. $\boldsymbol{\lambda}\beta\eta \vdash W_1 \longrightarrow W_2$ *implies* $W_2 \in cps(Values)$.
3. $\boldsymbol{\lambda}\beta\eta \vdash K_1 \longrightarrow K_2$ *implies* $K_2 \in cps(EvCont)$.

Proof. The proof is by induction on the structure of the terms:

1. Let $P_1 \in cps(\Lambda)$ and assume $\boldsymbol{\lambda}\beta\eta \vdash P_1 \longrightarrow P_2$. By definition, P_1 must be of the form $(K_1\ W_1)$ with $K_1 \in cps(EvCont)$ and $W_1 \in cps(Values)$. Three kinds of reductions are possible:
   - $\boldsymbol{\lambda}\beta\eta \vdash (K_1\ W_1) \longrightarrow (K_2\ W_1)$ because K_1 reduces to K_2. By the inductive hypothesis, $K_2 \in cps(EvCont)$ and therefore $P_2 \in cps(\Lambda)$.
   - $\boldsymbol{\lambda}\beta\eta \vdash (K_1\ W_1) \longrightarrow (K_1\ W_2)$ because W_1 reduces to W_2. The result follows also by the inductive hypothesis.
   - $\boldsymbol{\lambda}\beta\eta \vdash ((\lambda x.P)\ W_1) \longrightarrow P[x := W_1]$ because $K_1 \equiv (\lambda x.P)$. By Lemma 5.12, $P[x := W_1] \in cps(\Lambda)$.
2. Let $W_1 \in cps(Values)$ and assume $\boldsymbol{\lambda}\beta\eta \vdash W_1 \longrightarrow W_2$. There are two cases:
   - $W_1 \equiv x$ which is impossible, and
   - $W_1 \equiv \lambda k.K_1$ where $K_1 \in cps(EvCont)$. Then, either:
     - $\boldsymbol{\lambda}\beta\eta \vdash \lambda k.K_1 \longrightarrow \lambda k.K_2$ because K_1 reduces to K_2. The result follows by the inductive hypothesis.
     - $\boldsymbol{\lambda}\beta\eta \vdash \lambda k.W_3 k \longrightarrow W_3$ because $K_1 \equiv (W_3\ k)$ and $W_3 \in cps(Values)$ by definition.
3. Let $K_1 \in cps(EvCont)$ and assume $\boldsymbol{\lambda}\beta\eta \vdash K_1 \longrightarrow K_2$. Then, either:
   - $K_1 \equiv k$ which is impossible, or
   - $K_1 \equiv \lambda x.P_1$ and there are two cases:
     - $\boldsymbol{\lambda}\beta\eta \vdash (\lambda x.P_1) \longrightarrow (\lambda x.P_2)$ because P_1 reduces P_2. The result follows by the inductive hypothesis.
     - $\boldsymbol{\lambda}\beta\eta \vdash (\lambda x.Kx) \longrightarrow K$ because $P_1 \equiv Kx$ and $K \in cps(EvCont)$ by definition.
   - $K_1 \equiv (W\ K)$ and there are three cases:
     - $\boldsymbol{\lambda}\beta\eta \vdash (W\ K) \longrightarrow (W_1\ K)$ because W reduces to W_1 and the result follows by the inductive hypothesis.
     - $\boldsymbol{\lambda}\beta\eta \vdash (W\ K) \longrightarrow (W\ K_3)$ because K reduces to K_3 and the result follows also by induction.
     - $\boldsymbol{\lambda}\beta\eta \vdash ((\lambda k.K_3)\ K) \longrightarrow K_3[k := K]$ because $W \equiv \lambda k.K_3$. The result follows by Lemma 5.12. ∎

Finally, the grammar is also closed under the relevant substitutions.

Lemma 5.12 *Let $P \in cps(\Lambda)$, $W \in cps(Values)$, $K_1, K_2 \in cps(EvCont)$. Then,*

1. $P[x := W] \in cps(\Lambda)$.
2. $K_1[k := K_2] \in cps(EvCont)$.

Proof Idea. By induction on the structure of the terms P and K_1. ∎

For the opposite implication, i.e., that $cps(\Lambda)$ is a subset of S, it suffices to show that every $P \in cps(\Lambda)$ is reachable from Λ via $\mathcal{C}_k$ and $\beta\eta$.

Lemma 5.13 *For all $P \in cps(\Lambda)$, there exists an $M \in \Lambda$ such $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow P$. For all $K \in cps(EvCont)$, there exists an $E \in EvCont$ such $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{K}_k[\![E]\!] \longrightarrow\!\!\!\!\!\rightarrow K$.*

Proof. The proof is by lexicographic induction on $\langle \tilde{G}, |G| \rangle$ where G is an element P of $cps(\Lambda)$ or an element K of $cps(EvCont)$, $\tilde{G}$ is the number of abstractions of the form $\lambda k.K$ in G, and $|G|$ is the number of variables (including binding occurrences) in G. The proof proceeds by case analysis on the possible elements of $cps(\Lambda)$ and $cps(EvCont)$:

1. $G \equiv (k\ W)$, then there are four cases:
   (a) $W \equiv x$: take $M = x$.
   (b) $W \equiv \lambda k.k$: take $M = \lambda x.x$.
   (c) $W \equiv \lambda k.W_1 K$: let $P_1 \equiv ((W_1 K)\ x)$, then $\tilde{P_1} < \tilde{G}$ because P_1 has one less abstraction of the form $\lambda k.K$ than G. Therefore, by the inductive hypothesis, there exists an M_1 such that $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M_1]\!] \longrightarrow\!\!\!\!\!\rightarrow P_1$. Take $M = \lambda x.M_1$.
   (d) $W \equiv \lambda k.\lambda x.P_1$: by the inductive hypothesis, P_1 is reachable from a term M_1. Take $M = \lambda x.M_1$.
2. $G \equiv ((x\ K)\ W)$: by the inductive hypothesis, K is reachable from an evaluation context E. By an argument similar to the the first case, W is reachable from a value V. Take $M = E[(x\ V)]$.
3. $G \equiv (((\lambda k.K_1)\ K_2)\ W)$: by the inductive hypothesis, K_2 is reachable from an evaluation context E_2. By repeating the argument for the first case, the values $\lambda k.K_1$ and W are reachable from V_1 and V respectively. Take $M = ((\lambda x.((\lambda y.E_2[(y\ x)])\ V_1))\ V)$. Then,

$$\begin{array}{lll}
 & \mathcal{C}_k[\![((\lambda x.((\lambda y.E_2[(y\ x)])\ V_1))\ V)]\!] & \\
\equiv & ((\lambda x.((\lambda y.((y\ \mathcal{K}_k[\![E_2]\!])\ x))\ \Phi[\![V_1]\!]))\ \Phi[\![V]\!]) & \\
\longrightarrow\!\!\!\!\!\rightarrow & ((\lambda x.((\lambda y.((y\ K_2)\ x))\ \lambda k.K_1))\ W) & \text{(induction)} \\
\longrightarrow & ((\lambda x.(((\lambda k.K_1)\ K_2)\ x))\ W) & (\beta) \\
\longrightarrow & (((\lambda k.K_1)\ K_2)\ W) & (\beta)
\end{array}$$

4. $G \equiv ((\lambda x.P_1)\ W)$: by the inductive hypothesis, there exists an M_1 that reaches P_1. By repeating the argument for the first case, there exists also a value V that reaches W. Take $M = ((\lambda x.M_1)\ V)$.

5. $G \equiv k$: take $E = [\,]$.

6. $G \equiv (x\ K_1)$: by the inductive hypothesis, there exists an E_1 that reaches K_1. Take $E = E_1[(x\ [\,])]$.

7. $G \equiv ((\lambda k.K_1)\ K_2)$: similarly to case 3, take $E = ((\lambda x.((\lambda y.E_2[(y\ x)])\ V_1))\ [\,])$.

8. $G \equiv (\lambda x.P_1)$: take $E = ((\lambda x.M_1)\ [\,])$ where M_1 reaches P_1 by the inductive hypothesis. ∎

6 An Inverse CPS Transformation

Based on the inductive definition of the CPS language, the specification of an "inverse" to the CPS transformation is almost straightforward: the source term corresponding to the application of a continuation K to a value W is simply $E[V]$ where E is the evaluation context that syntactically represents the continuation K and V is the value that corresponds to W. The definition of the function $\mathcal{C}^{-1}$ (*un*-CPS) uses two auxiliary functions to translate continuations to evaluation contexts and values in the CPS language to values in the source language. Both definitions are straightforward.

Definition 6.1. $(\mathcal{C}^{-1}, \Phi^{-1}, \mathcal{K}^{-1})$

$$\begin{aligned}
\mathcal{C}^{-1} : cps(\Lambda) &\rightarrow \Lambda \\
\mathcal{C}^{-1}[\![(K\ W)]\!] &= \mathcal{K}^{-1}[\![K]\!][\Phi^{-1}[\![W]\!]]
\end{aligned}$$

$$\begin{aligned}
\Phi^{-1}[\![x]\!] &= x \\
\Phi^{-1}[\![(\lambda k.k)]\!] &= \lambda x.x \\
\Phi^{-1}[\![(\lambda k.WK)]\!] &= \lambda x.\mathcal{C}^{-1}[\![(W\ K)\ x]\!] \\
\Phi^{-1}[\![(\lambda k.\lambda x.P)]\!] &= \lambda x.\mathcal{C}^{-1}[\![P]\!]
\end{aligned}$$

$$\begin{aligned}
\mathcal{K}^{-1}[\![k]\!] &= [\,] \\
\mathcal{K}^{-1}[\![(x\ K)]\!] &= \mathcal{K}^{-1}[\![K]\!][(x\ [\,])] \\
\mathcal{K}^{-1}[\![((\lambda k.K_1)\ K_2)]\!] &= \mathcal{K}^{-1}[\![K_1[k := K_2]]\!] \\
\mathcal{K}^{-1}[\![(\lambda x.P)]\!] &= ((\lambda x.\mathcal{C}^{-1}[\![P]\!])\ [\,])
\end{aligned}$$

∎

Intuitively, $\mathcal{C}^{-1}$, Φ^{-1}, and $\mathcal{K}^{-1}$ are the "inverses" of $\mathcal{C}$, Φ, and $\mathcal{K}$ respectively. Moreover, $\mathcal{C}^{-1}$, Φ^{-1}, and $\mathcal{K}^{-1}$ apply to the syntactic categories $cps(\Lambda)$, $cps(\mathit{Values})$, and $cps(\mathit{EvCont})$ respectively and yield terms in the appropriate syntactic categories in the source language. To facilitate proofs of subsequent theorems, we first show that the output of $\mathcal{C}^{-1}$ is a proper subset of Λ.

Definition 6.2. (*Output of* $\mathcal{C}^{-1}$)

$$\begin{array}{rcll} M & ::= & E[V] & (\Lambda_u) \\ V & ::= & x \mid (\lambda x.M) & (\mathit{Values}_u) \\ E & ::= & [\,] \mid ((\lambda x.M)\ [\,]) \mid E[(x\ [\,])] & (\mathit{EvCont}_u) \end{array}$$

∎

Lemma 6.3 *Let* $P \in cps(\Lambda)$, $K \in cps(EvCont)$. *Then,* $\mathcal{C}^{-1}[\![P]\!] \in \Lambda_u$ *and* $\mathcal{K}^{-1}[\![K]\!] \in EvCont_u$.

Proof. The proof is by lexicogaphic induction on the number of abstractions of the form $\lambda k.K$ and the size of the terms (cpm. the proof of Lemma 5.13). It proceeds by case analysis on the possible inputs to $\mathcal{C}^{-1}$ and $\mathcal{K}^{-1}$.

- $P \equiv (K\ W)$, then $\mathcal{C}^{-1}[\![P]\!] \equiv \mathcal{K}^{-1}[\![K]\!][\Phi^{-1}[\![W]\!]]$. By induction, $\mathcal{K}^{-1}[\![K]\!] \in EvCont_u$. It remains to establish that $\Phi^{-1}[\![W]\!] \in Values_u$.
  - $W \equiv x$, then $\Phi^{-1}[\![W]\!] \equiv x \in Values_u$.
  - $W \equiv \lambda k.k$, then $\Phi^{-1}[\![W]\!] \equiv \lambda x.x$. Since $x \equiv [\,][x]$, then $\lambda x.x \in Values_u$.
  - $W \equiv \lambda k.WK$, then $\Phi^{-1}[\![W]\!] \equiv \lambda x.\mathcal{C}^{-1}[\![((WK)\ x)]\!]$. Because the term $((WK)\ x)$ has one less abstraction of the form $\lambda k.K$ than W, the inductive hypothesis applies to it. Therefore, $\mathcal{C}^{-1}[\![((WK)\ x)]\!] \in \Lambda_u$ which shows that $\Phi^{-1}[\![W]\!] \in Values_u$.
  - $W \equiv \lambda k.\lambda x.P$, then $\Phi^{-1}[\![W]\!] \equiv \lambda x.\mathcal{C}^{-1}[\![P]\!]$, then the result follows by induction.
- $K \equiv k$, then $\mathcal{K}^{-1}[\![K]\!] \equiv [\,] \in EvCont_u$.
- $K \equiv (x\ K_1)$, then $\mathcal{K}^{-1}[\![K]\!] \equiv \mathcal{K}^{-1}[\![K_1]\!][(x\ [\,])]$. The result is immediate because $\mathcal{K}^{-1}[\![K_1]\!] \in EvCont_u$ by induction.
- $K \equiv ((\lambda k.K_1)\ K_2)$, then $\mathcal{K}^{-1}[\![K]\!] \equiv \mathcal{K}^{-1}[\![K_1[k := K_2]]\!]$. Because k occurs exactly once in K_1, then term $K_1[k := K_2]$ has one less abstraction of the $\lambda k.K$ than $((\lambda k.K_1)\ K_2)$. Therefore, $\mathcal{K}^{-1}[\![K]\!] \in EvCont_u$ by the inductive hypothesis.
- $K \equiv \lambda x.P$, then $\mathcal{K}^{-1}[\![K]\!] \equiv ((\lambda x.\mathcal{C}^{-1}[\![P]\!])\ [\,])$ and the result follows by induction. ∎

For two distinct reasons, $\mathcal{C}^{-1}$ cannot be a complete inverse of $\mathcal{C}_k$. First, some CPS terms are the image of more than one source term. Second, some CPS terms are not the image of any source term. The first fact is a property of the function $\mathcal{C}_k$ that reduces administrative redexes on the fly. The second one is due to the closure of the set of CPS terms under $\beta\eta$-reductions. We discuss each point in detail below.[5]

[5] Danvy and Lawall [3, 6] define a *direct style transformation* mapping CPS programs into source terms. The transformation is the inverse of the Danvy-Filinski CPS transformation [5]; it is only applicable to images of Λ terms. For example, let

$$M = ((\lambda d.5)\ ((\lambda x.xx)\ (\lambda x.xx))).$$

The effect of administrative reductions: The function $\mathcal{C}_k$ incorporates the reduction of all administrative redexes from the output of the Fischer CPS. Hence, if $\mathcal{F}[\![M]\!]$ and $\mathcal{F}[\![N]\!]$ reduce to a common term by administrative reductions only, $\mathcal{C}_k[\![M]\!]$ is *identical* to $\mathcal{C}_k[\![N]\!]$. The definition of the function $\mathcal{C}_k$ shows that, in two cases, different inputs are indeed mapped to the same output.

- The first equivalence is:

  $$\mathcal{C}_k[\![E[((\lambda x.M)\ N)]]\!] \equiv \mathcal{C}_k[\![((\lambda x.E[M])\ N)]\!].$$

  The equation illustrates how the CPS transformation uses its knowledge about the continuation of an application. As indicated in Section 4, it "lifts" the application to top level and merges the continuation with the body of the application.

- The second equivalence is:

  $$\mathcal{C}_k[\![E[((z\ N)\ L)]]\!] \equiv \mathcal{C}_k[\![((\lambda x.E[(x\ L)])\ (z\ N))]\!].$$

  This equation captures another essential element of CPS transformations. According to folklore in the functional compiler-building community [4], the first aspect of a CPS transformation is to give the value of every application a name. In the above equation, the argument to $\mathcal{C}_k$ in the right hand side is a "flattened" version of the left hand side in which the nested application $(z\ N)$ is factored out and given a name.

In summary, we define two reductions on Λ that capture the effect of the administrative reductions performed by $\mathcal{C}_k$:

$$\begin{array}{rcllr} E[((\lambda x.M)\ N)] & \longrightarrow & ((\lambda x.E[M])\ N) & x \notin FV(E), E \neq [\,] & (\beta_{\mathit{lift}}) \\ E[((z\ N)\ L)] & \longrightarrow & ((\lambda x.E[(x\ L)])\ (z\ N)) & x \notin FV(E, L) & (\beta_{\mathit{flat}}) \end{array}$$

The reductions β_{lift} and β_{flat} define equivalence classes of source terms that map to the same CPS term. The function $\mathcal{C}^{-1}$ maps this CPS term to a particular representative of the equivalence class: the element in $\beta_{\mathit{lift}}\beta_{\mathit{flat}}$-normal form.

Lemma 6.4 *Let $P \in cps(\Lambda)$. Then, $\mathcal{C}^{-1}[\![P]\!]$ is in $\beta_{\mathit{lift}}\beta_{\mathit{flat}}$-normal form.*

Proof. It suffices to show that all terms generated by the grammar in Definition 6.2 are in $\beta_{\mathit{lift}}\beta_{\mathit{flat}}$-normal form. The proof is by induction on the structure of the terms. ∎

It follows that $\mathcal{C}^{-1}$ is an inverse of $\mathcal{C}_k$ on the subset of source terms in $\beta_{\mathit{lift}}\beta_{\mathit{flat}}$-normal form.

Then, according to Danvy/Filinski,

$$CPS(M) = ((\lambda xk.xxk)\ (\lambda xk.xxk)\ (\lambda \nu.((\lambda dk.k5)\ \nu\ (\lambda v.kv)))).$$

The latter expression reduces by two meaning-preserving β-reductions to:

$$((\lambda xk.xxk)\ (\lambda xk.xxk)\ (\lambda \nu.k5))$$

which is still a diverging term. The direct style transformation *cannot* be applied to this term because in the continuation $\lambda\nu.k5$, ν does not occur exactly once in $k5$. An extension of the direct style transformation that handles the above term fails to be an inverse of the Danvy/Filinski CPS transformation.

Theorem 6.5 *Let $M \in \Lambda$, $P \in cps(\Lambda)$, $E \in EvCont$, and $K \in cps(EvCont)$. Then,*

1. $\boldsymbol{\lambda}\beta_{\mathit{lift}}\beta_{\mathit{flat}} \vdash M \longrightarrow\!\!\!\!\rightarrow (\mathcal{C}^{-1} \circ \mathcal{C}_k)[\![M]\!]$ *and* $\boldsymbol{\lambda}\beta_{\mathit{lift}}\beta_{\mathit{flat}} \vdash E \longrightarrow\!\!\!\!\rightarrow (\mathcal{K}^{-1} \circ \mathcal{K}_k)[\![E]\!]$;
2. $(\mathcal{C}^{-1} \circ \mathcal{C}_k)[\![M]\!] \equiv M$ *for* $M = \mathcal{C}^{-1}[\![P]\!]$ *and* $(\mathcal{K}^{-1} \circ \mathcal{K}_k)[\![E]\!] \equiv E$ *for* $E = \mathcal{K}^{-1}[\![K]\!]$.

Proof. We prove the first claim by induction on the size of the argument to $\mathcal{C}_k$ or $\mathcal{K}_k$. The proof proceeds by cases on the possible inputs to the two functions:

1. $M \equiv V$: then there are two sub-cases:
   (a) $V \equiv x$: then x is identical to $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![x]\!]]\!]$.
   (b) $V \equiv \lambda x.N$: then by the inductive hypothesis $\lambda x.N$ reduces to $\lambda x.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![N]\!]]\!] \equiv \mathcal{C}^{-1}[\![\mathcal{C}_k[\![\lambda x.N]\!]]\!]$.
2. $M \equiv E[(x\ V)]$: then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv \mathcal{K}^{-1}[\![\mathcal{K}_k[\![E]\!]]\!][(x\ \Phi^{-1}[\![\Phi[\![V]\!]]\!])]$. If V is not a variable, then $|E| < |E[(x\ V)]|$. Therefore, by the inductive hypothesis, E reduces to $\mathcal{K}^{-1}[\![\mathcal{K}_k[\![E]\!]]\!]$, and by inlining the arguments for the first case, V reduces to $\Phi^{-1}[\![\Phi[\![V]\!]]\!]$. Otherwise, if V is a variable y, there are four cases:
   (a) $M \equiv (x\ y)$, then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv M$.
   (b) $M \equiv E_1[(z\ (x\ y))]$, then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv \mathcal{K}^{-1}[\![\mathcal{K}_k[\![E_1]\!]]\!][(z\ (x\ y))]$ and the result follows by the inductive hypothesis.
   (c) $M \equiv E_1[((\lambda z.L)\ (x\ y))]$, then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv ((\lambda z.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[L]]\!]]\!])\ (x\ y))$, and $E_1[((\lambda z.L)\ (x\ y))]$ reduces to $((\lambda z.E_1[L])\ (x\ y))$ by a β_{lift}-reduction. The latter term reduces to $((\lambda z.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[L]]\!]]\!])\ (x\ y))$ by induction.
   (d) $M \equiv E_1[((x\ y)\ L)]$, then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv ((\lambda u.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[(u\ L)]]\!]]\!])\ (x\ y))$ and by a β_{flat}-reduction, $E_1[((x\ y)\ L)]$ reduces to $((\lambda u.E_1[(u\ L)])\ (x\ y))$, which in turn reduces to $((\lambda u.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[(u\ L)]]\!]]\!])\ (x\ y))$ by induction.
3. $M \equiv E[((\lambda x.N)\ V)]$: then $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!] \equiv ((\lambda x.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E[N]]\!]]\!])\ \Phi^{-1}[\![\Phi[\![V]\!]]\!])$, and $E[((\lambda x.N)\ V)] \longrightarrow ((\lambda x.E[N])\ V)$ by β_{lift}, which reduces to $((\lambda x.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E[N]]\!]]\!])\ V)$ by the inductive hypothesis. The result follows because V reduces to $\Phi^{-1}[\![\Phi[\![V]\!]]\!]$ as in case 1.
4. $E = [\ ]$: then $\mathcal{K}^{-1}[\![\mathcal{K}_k[\![[\ ]]\!]]\!]$ is identical to $[\ ]$.
5. $E = E_1[(x\ [\ ])]$: then $\mathcal{K}^{-1}[\![\mathcal{K}_k[\![E]\!]]\!] \equiv \mathcal{K}^{-1}[\![\mathcal{K}_k[\![E_1]\!]]\!][(x\ [\ ])]$. The result follows by the inductive hypothesis.
6. $E = E_1[((\lambda x.N)\ [\ ])]$: similar to case 3.
7. $E = E_1[([\ ]\ N)]$: then $\mathcal{K}^{-1}[\![\mathcal{K}_k[\![E]\!]]\!] \equiv ((\lambda u.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[(u\ N)]]\!]]\!])\ [\ ])$. By a β_{flat}-reduction, $E_1[([\ ]\ N)] \longrightarrow ((\lambda u.E_1[(u\ N)])\ [\ ])$. By the inductive hypothesis, the latter term reduces to $((\lambda u.\mathcal{C}^{-1}[\![\mathcal{C}_k[\![E_1[(u\ N)]]\!]]\!])\ [\ ])$.

The proof of the second part differs slightly from the above. According to the grammar of Definition 6.2, cases 2d and 7 are impossible. Similarly, in cases 2c, 3 and 6 the surrounding context must be empty. Since these changes account for all the reductions from M to $\mathcal{C}^{-1}[\![\mathcal{C}_k[\![M]\!]]\!]$, $M \equiv \mathcal{C}^{-1}[\![P]\!]$. ∎

The closure of the set of CPS terms under $\beta\eta$-reductions: Because of arbitrary $\beta\eta$-reductions, the set of CPS terms includes terms that are not the image of any source term. For example, the following η-reduction generates such a term:

$$\mathcal{C}_k[\![\lambda x.x]\!] = \lambda k.\lambda x.kx \longrightarrow \lambda k.k$$

The function $\mathcal{C}^{-1}$ (conceptually) coerces $\lambda k.k$ first to $\lambda k.\lambda x.kx$, i.e.,

$$\mathcal{C}^{-1}[\![\lambda k.k]\!] = \mathcal{C}^{-1}[\![\lambda k.\lambda x.kx]\!] = \lambda x.x.$$

Thus, P is generally not identical to $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![P]\!]$. However, $\mathcal{C}_k$ is naturally the inverse of $\mathcal{C}^{-1}$ on the subset of CPS terms that are images of source terms.

Theorem 6.6 *Let $P \in cps(\Lambda)$, $M \in \Lambda$, $K \in cps(EvCont)$, and $E \in EvCont$. Then,*

1. $\boldsymbol{\lambda}\beta\eta \vdash (\mathcal{C}_k \circ \mathcal{C}^{-1})[\![P]\!] = P$ *and* $\boldsymbol{\lambda}\beta\eta \vdash (\mathcal{K}_k \circ \mathcal{K}^{-1})[\![K]\!] = K$*;*
2. $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![P]\!] \equiv P$ *for* $P = \mathcal{C}_k[\![M]\!]$ *and* $(\mathcal{K}_k \circ \mathcal{K}^{-1})[\![K]\!] \equiv K$ *for* $K \equiv \mathcal{K}_k[\![E]\!]$.

Proof Sketch. The proof of the first claim is similar to the proof of Lemma 5.13. We proceed by case analysis on the possible elements of $cps(\Lambda)$ or $cps(Evcont)$:

1. $G \equiv (k\ W)$, then there are four cases:
   (a) $W \equiv x$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv (k\ x) \equiv G$.
   (b) $W \equiv \lambda k.k$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv (k\ \lambda k.\lambda x.kx)$, which reduces to G by an η-reduction.
   (c) $W \equiv \lambda k.W_1 K$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv (k\ \lambda k.\lambda x.\mathcal{C}_k[\![\mathcal{C}^{-1}[\![((W_1\ K)\ x)]\!]]\!])$ which is provably equal to $(k\ \lambda k.\lambda x.((W_1\ K)\ x))$ by the inductive hypothesis. The latter term reduces to G by an η-reduction.
   (d) $W \equiv \lambda k.\lambda x.P_1$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv (k\ \lambda k.\lambda x.\mathcal{C}_k[\![\mathcal{C}^{-1}[\![P_1]\!]]\!])$ and the result follows by the inductive hypothesis.

2. $G \equiv ((x\ K)\ W)$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] = ((x\ \mathcal{K}_k[\![\mathcal{K}^{-1}[\![K]\!]]\!])\ \Phi[\![\Phi^{-1}[\![W]\!]]\!])$. By the inductive hypothesis $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{K}_k[\![\mathcal{K}^{-1}[\![K]\!]]\!] = K$ and by argument similar the first case $\boldsymbol{\lambda}\beta\eta \vdash \Phi[\![\Phi^{-1}[\![W]\!]]\!] = W$.

3. $G \equiv (((\lambda k.K_1)\ K_2)\ W)$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv (\mathcal{K}_k[\![\mathcal{K}^{-1}[\![K_1[k := K_2]]\!]]\!]\ \Phi[\![\Phi^{-1}[\![W]\!]]\!])$. Because k occurs exactly once in K_1, then $K_1[k := K_2]$ has one less abstraction of the form $\lambda k.K$ than $((\lambda k.K_1)\ K_2)$. Therefore, by the inductive hypothesis, $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{K}_k[\![\mathcal{K}^{-1}[\![K_1[k := K_2]]\!]]\!] = K_1[k := K_2]$. By repeating the argument for the first case, $\Phi[\![\Phi^{-1}[\![W]\!]]\!]$ and W are also provably equal. It follows that $\boldsymbol{\lambda}\beta\eta \vdash (K_1[k := K_2]\ W) = \mathcal{C}_k[\![\mathcal{C}^{-1}[\![(((\lambda k.K_1)\ K_2)\ W)]\!]]\!]$. The result follows by a β-reduction.

4. $G \equiv ((\lambda x.P_1)\ W)$: Then $(\mathcal{C}_k \circ \mathcal{C}^{-1})[\![G]\!] \equiv ((\lambda x.\mathcal{C}_k[\![\mathcal{C}^{-1}[\![P_1]\!]]\!])\ \Phi[\![\Phi^{-1}[\![W]\!]]\!])$. By the inductive hypothesis, $\boldsymbol{\lambda}\beta\eta \vdash P_1 = \mathcal{C}_k[\![\mathcal{C}^{-1}[\![P_1]\!]]\!]$. By the argument in case 1, $\boldsymbol{\lambda}\beta\eta \vdash W = \Phi[\![\Phi^{-1}[\![W]\!]]\!]$. Therefore, $\boldsymbol{\lambda}\beta\eta \vdash ((\lambda.P_1)\ W) = ((\lambda x.\mathcal{C}_k[\![\mathcal{C}^{-1}[\![P_1]\!]]\!])\ \Phi[\![\Phi^{-1}[\![W]\!]]\!])$.

5. $G \equiv k$: Then $(\mathcal{K}_k \circ \mathcal{K}^{-1})[\![G]\!] \equiv G$.

6. $G \equiv (x\ K_1)$: Then $(\mathcal{K}_k \circ \mathcal{K}^{-1})[\![G]\!] \equiv (x\ \mathcal{K}_k[\![\mathcal{K}^{-1}[\![K_1]\!]]\!])$ and the result follows by the inductive hypothesis.

7. $G \equiv ((\lambda k.K_1)\ K_2)$: Then $(\mathcal{K}_k \circ \mathcal{K}^{-1})[\![G]\!] \equiv \mathcal{K}_k[\![\mathcal{K}^{-1}[\![K_1[k := K_2]]\!]]\!]$ and this case is simliar to case 3.

8. $G \equiv (\lambda x.P_1)$: Then $(\mathcal{K}_k \circ \mathcal{K}^{-1})[\![G]\!] \equiv (\lambda x.\mathcal{C}_k[\![\mathcal{C}^{-1}[\![P_1]\!]]\!])$ and the result follows by induction.

The proof of the second part is identical to the above but it excludes the cases that do not correspond to images of source terms. In particular, it excludes cases 1b and 1c because, in the image of a source term, the body of a $\lambda k.K$ abstraction must be of the form $\lambda x.P$. Moreover, it excludes cases 3 and 7 because they contain the administrative redex $((\lambda k.K_1)\ K_2)$. ∎

7 Completeness and Soundness

Using the partial inverse of the CPS transformation, we can systematically derive a set of additional axioms A for $\boldsymbol{\lambda}_v$ such that $\boldsymbol{\lambda}_v A$ is complete for $\beta\eta$ reasoning about CPS programs. Once we have the new axiom set, we prove its soundness in the second subsection. In the last subsection, we briefly discuss the connection to Moggi's computational λ-calculus.

7.1 Completeness

As specified in Section 5, the possible β- and η-reductions on CPS terms are $\beta_w, \beta_k, \eta_w, \eta_k$. We first outline the derivation of reductions corresponding to η_k. Let $(\lambda x.Kx) \longrightarrow K$ where $x \notin FV(K)$. Applying $\mathcal{K}^{-1}$ to both sides of the reduction, we get:

$$((\lambda x.\mathcal{C}^{-1}[\![Kx]\!])\ [\ ]) \qquad \text{and} \qquad \mathcal{K}^{-1}[\![K]\!].$$

To understand how the left hand side could reduce to the right hand side, we proceed by case analysis on K:

- $K \equiv k$: The reduction becomes $((\lambda x.x)\ [\ ]) \longrightarrow [\ ]$. Since the empty context generally stands for an arbitrary expression, A should therefore contain the reduction:

$$((\lambda x.x)\ M) \ \longrightarrow\ M \qquad (\beta_{id})$$

- $K \equiv (y\ K_1)$: The reduction becomes $((\lambda x.\mathcal{K}^{-1}[\![K_1]\!][(y\ x)])\ [\ ]) \longrightarrow \mathcal{K}^{-1}[\![K_1]\!][(y\ [\ ])]$. By a similar argument as above, we must add the following reduction to A:

$$((\lambda x.E[(y\ x)])\ M) \ \longrightarrow\ E[(y\ M)] \qquad (\beta_\Omega)$$

- $K \equiv ((\lambda k.K_1)\ K_2)$: The reduction becomes:

$$((\lambda x.\mathcal{C}^{-1}[\![K_1[k := K_2]\ x]\!])\ [\ ]) \longrightarrow \mathcal{K}^{-1}[\![K_1[k := K_2]]\!].$$

  Since the term $(K_1[k := K_2]\ x)$ has one less $\lambda k.K$ abstraction than K, the inductive hypothesis provides an appropriate equivalence.

- $K \equiv \lambda y.P$: By an β_v-reduction, the left hand side $((\lambda x.((\lambda y.\mathcal{C}^{-1}[\![P]\!])\ x))\ [\ ])$ reduces to $((\lambda x.\mathcal{C}^{-1}[\![P]\!][y := x])\ [\ ])$, which is identical to the right hand side.

The cases for the other reductions on CPS terms are similar. The resulting set of source reductions, A, includes all the previously derived reductions and η_v: see Figure 1.

$$
\begin{array}{rcll r}
((\lambda x.M)\ V) & \longrightarrow & M[x := V] & & (\beta_v) \\
(\lambda x.Vx) & \longrightarrow & V & x \notin FV(V) & (\eta_v) \\
E[((\lambda x.M)\ N)] & \longrightarrow & ((\lambda x.E[M])\ N) & x \notin FV(E), E \neq [\ ] & (\beta_{lift}) \\
E[((z\ N)\ L)] & \longrightarrow & ((\lambda x.E[(x\ L)])\ (z\ N)) & x \notin FV(E, L) & (\beta_{flat}) \\
((\lambda x.x)\ M) & \longrightarrow & M & & (\beta_{id}) \\
((\lambda x.E[(y\ x)])\ M) & \longrightarrow & E[(y\ M)] & x \notin FV(E[y]) & (\beta_\Omega)
\end{array}
$$

Figure 1: Source Reductions: $A \stackrel{df}{=} \{\eta_v, \beta_{lift}, \beta_{flat}, \beta_{id}, \beta_\Omega\}$

The Completeness Lemma summarizes the connection between the notions of reductions on $cps(\Lambda)$ and the new reductions.

Lemma 7.1 (Completeness) *Let $P \in cps(\Lambda)$.*

$$
\begin{array}{llllll}
1.\ \textit{If} & \boldsymbol{\lambda}\eta_k \vdash P \longrightarrow Q & \textit{then} & \boldsymbol{\lambda}\beta_v\beta_{id}\beta_\Omega & \vdash \mathcal{C}^{-1}[\![P]\!] & \twoheadrightarrow \mathcal{C}^{-1}[\![Q]\!] \\
2.\ \textit{If} & \boldsymbol{\lambda}\beta_k \vdash P \longrightarrow Q & \textit{then} & & \mathcal{C}^{-1}[\![P]\!] & \equiv \mathcal{C}^{-1}[\![Q]\!] \\
3.\ \textit{If} & \boldsymbol{\lambda}\eta_w \vdash P \longrightarrow Q & \textit{then} & \boldsymbol{\lambda}\eta_v & \vdash \mathcal{C}^{-1}[\![P]\!] & \twoheadrightarrow \mathcal{C}^{-1}[\![Q]\!] \\
4.\ \textit{If} & \boldsymbol{\lambda}\beta_w \vdash P \longrightarrow Q & \textit{then} & \boldsymbol{\lambda}\beta_v\beta_{lift}\beta_{id}\beta_\Omega & \vdash \mathcal{C}^{-1}[\![P]\!] & \twoheadrightarrow \mathcal{C}^{-1}[\![Q]\!]
\end{array}
$$

Proof. The proof of each case is distinct.

1. η_k-reduction: The proof is outlined at the beginning of the section.
2. β_k-reduction: By the definition of $\mathcal{K}^{-1}$, $\mathcal{K}^{-1}[\![((\lambda k.K_1)\ K_2)]\!] \equiv \mathcal{K}^{-1}[\![K_1[k := K_2]]\!]$.
3. η_w-reduction: Applying Φ^{-1} to both sides of the reduction, we get $\Phi^{-1}[\![(\lambda k.Wk)]\!] \equiv (\lambda x.\mathcal{C}^{-1}[\![((Wk)\ x)]\!])$ and $\Phi^{-1}[\![W]\!]$. The possible cases are:
   - $W \equiv z$: then the reduction becomes the η_v-reduction: $(\lambda x.zx) \longrightarrow x$.
   - $W \equiv \lambda k.k$: then both sides of the reduction are identical.
   - $W \equiv \lambda k.W_1K$: then again both sides of the reduction are identical.
   - $W \equiv \lambda k.\lambda z.P$, then $\lambda x.((\lambda z.\mathcal{C}^{-1}[\![P]\!])\ x) \longrightarrow \lambda z.\mathcal{C}^{-1}[\![P]\!]$ is an η_v-reduction.
4. β_w-reduction: Here, we need to show that:

   $$\boldsymbol{\lambda}\beta_v\beta_{lift}\beta_{id}\beta_\Omega \vdash \mathcal{C}^{-1}[\![((\lambda x.P)\ W)]\!] \twoheadrightarrow \mathcal{C}^{-1}[\![P[x := W]]\!].$$

   The left hand side of the reduction is equivalent to $((\lambda x.\mathcal{C}^{-1}[\![P]\!])\ \Phi^{-1}[\![W]\!])$, which in turn reduces to $\mathcal{C}^{-1}[\![P]\!][x := \Phi^{-1}[\![W]\!]]$ by a β_v-reduction. The latter term reduces to $\mathcal{C}^{-1}[\![P[x := W]]\!]$ by the Substitution Lemma below. ∎

In the above theorem, the proofs of the first three cases are complete. Only claim 4 requires the Substitution Lemma, which in turn refers back to Theorem 7.1 (case 1).

Lemma 7.2 (Substitution) *Let $P \in cps(\Lambda)$, $W \in cps(Values)$, $K \in cps(EvCont)$. Then,*

1. $\boldsymbol{\lambda}\beta_v\beta_{lift}\beta_{id}\beta_\Omega \vdash \mathcal{C}^{-1}[\![P]\!][x := \Phi^{-1}[\![W]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}^{-1}[\![P[x := W]]\!]$.
2. $\boldsymbol{\lambda}\beta_v\beta_{lift}\beta_{id}\beta_\Omega \vdash \mathcal{K}^{-1}[\![K]\!][x := \Phi^{-1}[\![W]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}^{-1}[\![P[x := W]]\!]$.

Proof. The proof is by lexicographic induction on $\langle \tilde{G}, |G| \rangle$, i.e., the number of abstractions of the form $\lambda k.K$ and the size of the terms. The proof proceeds by cases on the arguments to $\mathcal{C}^{-1}$ and $\mathcal{K}^{-1}$.

- $P \equiv (K\ W_1)$: then, $\mathcal{C}^{-1}[\![P]\!][x := \Phi^{-1}[\![W]\!]] \equiv \mathcal{K}^{-1}[\![K]\!][\Phi^{-1}[\![W_1]\!]]\ [x := \Phi^{-1}[\![W]\!]]$, which by the inductive hypothesis reduces to $\mathcal{K}^{-1}[\![K[x := W]]\!][\Phi^{-1}[\![W_1]\!][x := \Phi^{-1}[\![W]\!]]]$. It remains to establish that substitution commutes with Φ^{-1} as well. There are five cases:

    1. $W_1 \equiv x$: then $\Phi^{-1}[\![x]\!][x := \Phi^{-1}[\![W]\!]] \equiv \Phi^{-1}[\![x[x := W]]\!]$.
    2. $W_1 \equiv z$ and $z \not\equiv x$: then the result is immediate.
    3. $W_1 \equiv \lambda k.k$: immediate since x is not free.
    4. $W_1 \equiv \lambda k.W_2 k$: then, $\Phi^{-1}[\![W_1]\!][x := \Phi^{-1}[\![W]\!]] \equiv \lambda z.\mathcal{C}^{-1}[\![((W_2 K)\ z)]\!][x := \Phi^{-1}[\![W]\!]]$. By induction, we get $\lambda z.\mathcal{C}^{-1}[\![(W_2 K)[x := W]\ z]\!]$. The latter term is identical to $\Phi^{-1}[\![(\lambda k.W_2 K)[x := W]]\!]$ as desired.
    5. $W_1 \equiv \lambda k.\lambda z.P_1$ ($z \not\equiv x$): then, $\Phi^{-1}[\![W_1]\!][x := \Phi^{-1}[\![W]\!]] \equiv (\lambda z.\mathcal{C}^{-1}[\![P]\!])[x := \Phi^{-1}[\![W]\!]]$. By induction, we get $\lambda z.\mathcal{C}^{-1}[\![P[x := W]]\!] \equiv \Phi^{-1}[\![(\lambda k.\lambda z.P)[x := W]]\!]$.

- $K \equiv k$: then the claim is vacuously true because $k \neq x$.

- $K \equiv ((\lambda k.K_1)\ K_2)$: here,

$$\mathcal{K}^{-1}[\![((\lambda k.K_1)\ K_2)]\!][x := \Phi^{-1}[\![W]\!]] \equiv \mathcal{K}^{-1}[\![K_1[k := K_2]]\!][x := \Phi^{-1}[\![W]\!]].$$

  By induction, the latter term reduces to $\mathcal{K}^{-1}[\![K_1[k := K_2][x := W]]\!]$, which is identical to $\mathcal{K}^{-1}[\![((\lambda k.K_1[x := W])\ K_2[x := W])]\!]$.

- $K \equiv \lambda z.P_1$ ($z \not\equiv x$): $\mathcal{K}^{-1}[\![\lambda z.P_1]\!][x := \Phi^{-1}[\![W]\!]] \equiv ((\lambda z.\mathcal{C}^{-1}[\![P]\!][x := \Phi^{-1}[\![W]\!]])\ [\,])$. By induction, $((\lambda z.\mathcal{C}^{-1}[\![P[x := W]]\!])\ [\,]) \equiv \mathcal{K}^{-1}[\![\lambda z.P[x := W]]\!]$.

- $K \equiv \lambda x.P_1$: immediate since x is not free.

- $K \equiv (z\ K_1)$ and $z \not\equiv x$: this is a special case of the next clause.

- $K \equiv (x\ K_1)$: then the left hand side is $\mathcal{K}^{-1}[\![K_1]\!][x := \Phi^{-1}[\![W]\!]][(\Phi^{-1}[\![W]\!]\ [\,])]$. By induction, the latter term reduces to $\mathcal{K}^{-1}[\![K_1[x := W]]\!]\ [(\Phi^{-1}[\![W]\!]\ [\,])]$. For readablity, let $K' \equiv K_1[x := W]$. The goal is to prove that

$$\mathcal{K}^{-1}[\![K']\!][(\Phi^{-1}[\![W]\!]\ [\,])] \text{ reduces to } \mathcal{K}^{-1}[\![(W\ K')]\!].$$

  We proceed by cases of W:

1. $W \equiv y$: then $\mathcal{K}^{-1}[\![K']\!][(y\ [\ ])] \equiv \mathcal{K}^{-1}[\![(y\ K')]\!]$.
2. $W \equiv \lambda k.k$: then, $\mathcal{K}^{-1}[\![K']\!][((\lambda x.x)\ [\ ])]$ reduces to $\mathcal{K}^{-1}[\![K']\!]$ by an β_{id}-reduction. The latter term is identical to $\mathcal{K}^{-1}[\![((\lambda k.k)\ K')]\!]$.
3. $W \equiv \lambda k.W_3K$: then, the left hand side $\mathcal{K}^{-1}[\![K']\!][((\lambda y.\mathcal{C}^{-1}[\![((W_3K)\ y)]\!])\ [\ ])] \equiv \mathcal{K}^{-1}[\![K']\!][\mathcal{K}^{-1}[\![(\lambda y.((W_3K)\ y))]\!]]$. By the first case in Lemma 7.1, this reduces to $\mathcal{K}^{-1}[\![K']\!][\mathcal{K}^{-1}[\![(W_3K)]\!]]$. Moreover, by Lemma 7.3, the latter term reduces to $\mathcal{K}^{-1}[\![(W_3K)[k := K']]\!]$, which is identical to $\mathcal{K}^{-1}[\![((\lambda k.W_3K)\ K')]\!]$.
4. $W \equiv \lambda k.\lambda y.P_2$: then by a β_{lift}-reduction, $\mathcal{K}^{-1}[\![K']\!][((\lambda z.\mathcal{C}^{-1}[\![P_2]\!])\ [\ ])]$ reduces to $((\lambda z.\mathcal{K}^{-1}[\![K']\!][\mathcal{C}^{-1}[\![P_2]\!]])\ [\ ])$. By Lemma 7.3, the latter term reduces to $((\lambda z.\mathcal{C}^{-1}[\![P_2[k := K']]\!])\ [\ ]) \equiv \mathcal{K}^{-1}[\![\lambda z.P_2[k := K']]\!] \equiv \mathcal{K}^{-1}[\![((\lambda k.\lambda z.P_2)\ K')]\!]$. ∎

Lemma 7.3 (Continuation) *Let $K, K_1, K_2 \in cps(EvCont)$, $P \in cps(\Lambda)$. Then,*

1. $\boldsymbol{\lambda}\beta_{lift} \vdash \mathcal{K}^{-1}[\![K]\!][\mathcal{C}^{-1}[\![P]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}^{-1}[\![P[k := K]]\!]$
2. $\boldsymbol{\lambda}\beta_{lift} \vdash \mathcal{K}^{-1}[\![K_2]\!][\mathcal{K}^{-1}[\![K_1]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}^{-1}[\![K_1[k := K_2]]\!]$

Proof. The proof is by induction on the number of abstractions of the $\lambda k.K$ and the size of P or K_1. It proceeds by case analysis on the possible elements of $cps(\Lambda)$ and $cps(EvCont)$.

- $P \equiv (K_3\ W)$, then $\mathcal{K}^{-1}[\![K]\!][\mathcal{C}^{-1}[\![P]\!]] \equiv \mathcal{K}^{-1}[\![K]\!][\mathcal{K}^{-1}[\![K_3]\!][\Phi^{-1}[\![W]\!]]]$. By the inductive hypothesis, $\boldsymbol{\lambda}\beta_{lift} \vdash \mathcal{K}^{-1}[\![K]\!][\mathcal{K}^{-1}[\![K_3]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}^{-1}[\![K_3[k := K]]\!]$. Therefore, $\boldsymbol{\lambda}\beta_{lift} \vdash \mathcal{K}^{-1}[\![K]\!][\mathcal{C}^{-1}[\![P]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}^{-1}[\![K_3[k := K]]\!][\Phi^{-1}[\![W]\!]]$. The latter term is identical to $\mathcal{K}^{-1}[\![(K_3\ W)[k := K]]\!]$ since k is never free in W.

- $K_1 \equiv k$: then both sides are identical to $\mathcal{K}^{-1}[\![K_2]\!]$.

- $K_1 \equiv (x\ K_3)$, then $\mathcal{K}^{-1}[\![K_2]\!][\mathcal{K}^{-1}[\![K_3]\!][(x\ [\ ])]]$ reduces to $\mathcal{K}^{-1}[\![K_3[k := K_2]]\!][(x\ [\ ])]$ by induction. The latter term is identical to $\mathcal{K}^{-1}[\![(x\ K_3[k := K_2])]\!]$.

- $K_1 \equiv ((\lambda k.K_3)\ K_4)$, then $\mathcal{K}^{-1}[\![K_2]\!][\mathcal{K}^{-1}[\![K_3[k := K_4]]\!]]$ reduces to $\mathcal{K}^{-1}[\![K_3[k := K_4][k := K_2]]\!]$ by induction. This term is identical to $\mathcal{K}^{-1}[\![K_3[k := K_4[k := K_2]]]\!]$, which in turn is identical to $\mathcal{K}^{-1}[\![((\lambda k.K_3)\ K_4)[k := K_2]]\!]$.

- $K_1 \equiv \lambda x.P$: then, $\mathcal{K}^{-1}[\![K_2]\!][((\lambda x.\mathcal{C}^{-1}[\![P]\!])\ [\ ])]$ reduces to $((\lambda x.\mathcal{K}^{-1}[\![K_2]\!][\mathcal{C}^{-1}[\![P]\!]])\ [\ ])$ by a β_{lift}-reduction. By induction, this term reduces to

$$((\lambda x.\mathcal{C}^{-1}[\![P[k := K_2]]\!])\ [\ ]) \equiv \mathcal{K}^{-1}[\![(\lambda x.P)[k := K_2]]\!]. \ ∎$$

The Completeness Theorem is a direct consequence of the above results.

Theorem 7.4 (Completeness) *If $\boldsymbol{\lambda}\beta\eta \vdash P \longrightarrow\!\!\!\!\!\rightarrow Q$ then $\boldsymbol{\lambda}_v A \vdash \mathcal{C}^{-1}[\![P]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}^{-1}[\![Q]\!]$.*

Proof. By pasting together the proofs of the Completeness Lemma. ∎

7.2 Soundness

The set of source reductions in Figure 1 is *sound* with respect to the equational theory over CPS terms. In other words, for a source reduction $M \longrightarrow N$, $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] = \mathcal{C}_k[\![N]\!]$. In fact, we can prove the stronger results of the following lemma.

Lemma 7.5 (Soundness) *Let* $M \in \Lambda$.

$$\begin{array}{llllllll}
1.\ \textit{If}\ \boldsymbol{\lambda}\beta_v & \vdash & M \longrightarrow N & \textit{then} & \boldsymbol{\lambda}\beta & \vdash & \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![N]\!] \\
2.\ \textit{If}\ \boldsymbol{\lambda}\eta_v & \vdash & M \longrightarrow N & \textit{then} & \boldsymbol{\lambda}\eta_w\eta_k & \vdash & \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![N]\!] \\
3.\ \textit{If}\ \boldsymbol{\lambda}\beta_{lift} & \vdash & M \longrightarrow N & \textit{then} & & & \mathcal{C}_k[\![M]\!] \equiv \mathcal{C}_k[\![N]\!] \\
4.\ \textit{If}\ \boldsymbol{\lambda}\beta_{flat} & \vdash & M \longrightarrow N & \textit{then} & & & \mathcal{C}_k[\![M]\!] \equiv \mathcal{C}_k[\![N]\!] \\
5.\ \textit{If}\ \boldsymbol{\lambda}\beta_{id} & \vdash & M \longrightarrow N & \textit{then} & \boldsymbol{\lambda}\eta_k & \vdash & \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![M]\!] \\
6.\ \textit{If}\ \boldsymbol{\lambda}\beta_\Omega & \vdash & M \longrightarrow N & \textit{then} & \boldsymbol{\lambda}\eta_k & \vdash & \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![N]\!]
\end{array}$$

Proof. The cases for β_{lift} and β_{flat} are immediate from the definition of the function $\mathcal{C}_k$. We present only the proof for β_v-reductions in Lemma 7.6. The other proofs are similar. ∎

Lemma 7.6 (β_v) $\boldsymbol{\lambda}\beta \vdash \mathcal{C}_k[\![((\lambda x.M)\ V)]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![M[x := V]]\!]$.

Proof. By definition of $\mathcal{C}_k$,

$$\mathcal{C}_k[\![((\lambda x.M)\ V)]\!] \equiv ((\lambda x.\mathcal{C}_k[\![M]\!])\ \Phi[\![V]\!]).$$

The latter term reduces to $\mathcal{C}_k[\![M]\!][x := \Phi[\![V]\!]]$. It remains to establish that substitution commutes with $\mathcal{C}_k$, i.e.,

1. $\boldsymbol{\lambda}\beta \vdash \mathcal{C}_k[\![M]\!][x := \Phi[\![V]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![M[x := V]]\!]$.
2. $\boldsymbol{\lambda}\beta \vdash \mathcal{K}_k[\![E]\!][x := \Phi[\![V]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}_k[\![E[x := V]]\!]$.

The proof is by induction on the size of the argument to $\mathcal{C}_k$ or $\mathcal{K}_k$. Except for one case, the inductive hypothesis applies immediately. The interesting case occurs when $M \equiv E[(x\ U)]$:

$$\begin{array}{rcl}
\mathcal{C}_k[\![E[(x\ U)]]\!][x := \Phi[\![V]\!]] & \equiv & ((x\ \mathcal{K}_k[\![E]\!])\ \Phi[\![U]\!])[x := \Phi[\![V]\!]] \\
& \longrightarrow & ((\Phi[\![V]\!]\ \mathcal{K}_k[\![E]\!][x := \Phi[\![V]\!]])\ \Phi[\![U]\!][x := \Phi[\![V]\!]]) \\
& \longrightarrow\!\!\!\!\!\rightarrow & ((\Phi[\![V]\!]\ \mathcal{K}_k[\![E[x := V]]\!])\ \Phi[\![U[x := V]]\!])
\end{array}$$

The last line follows by cases on E if U is a variable. Otherwise, it follows by the inductive hypothesis. For readability, let $E' \equiv E[x := V]$ and $U' \equiv U[x := V]$. The goal is to prove that:

$$((\Phi[\![V]\!]\ \mathcal{K}_k[\![E']\!])\ \Phi[\![U']\!]) \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![E'[(V\ U')]]\!]$$

We proceed by cases of V:

- $V \equiv z$, then both sides are identical.
- $V \equiv \lambda z.L$, then by a β-reduction,

  $$(((\lambda k.\lambda z.\mathcal{C}_k[\![L]\!])\ \mathcal{K}_k[\![E']\!])\ \Phi[\![U']\!]) \longrightarrow ((\lambda z.\mathcal{C}_k[\![L]\!][k := \mathcal{K}_k[\![E']\!]])\ \Phi[\![U']\!]).$$

  By Lemma 7.7, the latter term reduces to $((\lambda z.\mathcal{C}_k[\![E'[L]]\!])\ \Phi[\![U']\!])$, which is identical to $\mathcal{C}_k[\![E'[((\lambda z.L)\ U')]]\!]$. ∎

Lemma 7.7 (Context) *Let $M \in \Lambda$, $E, E_1 \in EvCont$. Then,*

$$\boldsymbol{\lambda}\beta \vdash \mathcal{C}_k[\![M]\!][k := \mathcal{K}_k[\![E]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![E[M]]\!] \text{ and } \boldsymbol{\lambda}\beta \vdash \mathcal{K}_k[\![E_1]\!][k := \mathcal{K}_k[\![E]\!]] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{K}_k[\![E[E_1]]\!]$$

.

Proof Idea. The proof is by induction on the size of M or E_1. ∎

The Soundness theorem is a direct consequence of these results.

Theorem 7.8 (Soundness) *If $\boldsymbol{\lambda}_v A \vdash M \longrightarrow\!\!\!\!\!\rightarrow N$ then $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] \longrightarrow\!\!\!\!\!\rightarrow \mathcal{C}_k[\![N]\!]$.*

7.3 Correspondence

Unfortunately, if $\mathcal{C}_k[\![M]\!]$ reduces to $\mathcal{C}_k[\![N]\!]$, M does not reduce to N unless N is in $\beta_{lift}\beta_{flat}$-normal form. Figure 2 summarizes the relationship between proofs on either side. The dotted lines correspond to the application of $\mathcal{C}_k$ or $\mathcal{C}^{-1}$. The solid lines represent sequences of reductions.

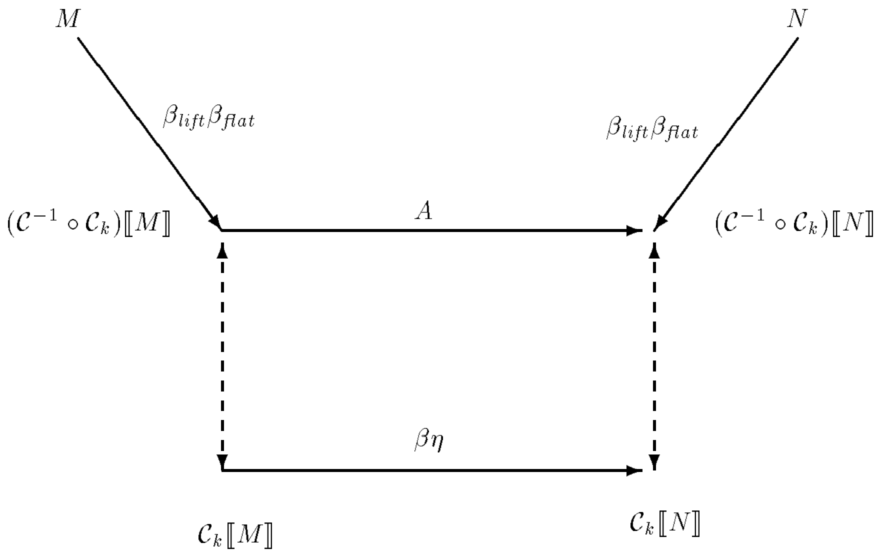

Figure 2: The correspondence Theorem.

Although the functions $\mathcal{C}_k$ and $\mathcal{C}^{-1}$ do not always preserve reductions, they preserve equality.

Theorem 7.9 (Correspondence) *The calculi $\boldsymbol{\lambda}\beta\eta$ and $\boldsymbol{\lambda}_v A$ are equivalent in the following sense:*

1. $\boldsymbol{\lambda}_v A \vdash M = (\mathcal{C}^{-1} \circ \mathcal{C}_k)[\![M]\!]$.
2. $\boldsymbol{\lambda}\beta\eta \vdash P = (\mathcal{C}_k \circ \mathcal{C}^{-1})[\![P]\!]$.

3. $\boldsymbol{\lambda}_v A \vdash M = N$ *iff* $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!] = \mathcal{C}_k[\![N]\!]$.

4. $\boldsymbol{\lambda}\beta\eta \vdash P = Q$ *iff* $\boldsymbol{\lambda}_v A \vdash \mathcal{C}^{-1}[\![P]\!] = \mathcal{C}^{-1}[\![Q]\!]$.

Note: Other CPS transformations. The correspondence theorem does not depend on any specific aspects of $\mathcal{C}_k$ or $\mathcal{F}$. Rather the result is valid for *any* CPS transformation *cps* that satisfies the following condition for $M \in \Lambda$:

$$\boldsymbol{\lambda}\beta\eta \vdash \mathcal{F}[\![M]\!] = cps(M).$$

End Note

7.4 The computational λ-calculus

The calculus $\boldsymbol{\lambda}_v A$ is equivalent to an untyped variant of Moggi's computational λ-calculus $\boldsymbol{\lambda}_c$ [21]. Specifically, if we ignore the types of expressions, eliminate product and computational expressions, re-interpret Moggi's **let**-expression as the usual abbreviation for a λ-application, and apply his **let**-axioms to the expanded expressions, then the basic reductions of $\boldsymbol{\lambda}_c$ are β_v, η_v, β_{id} plus the additional reductions of Figure 3.

$$\begin{array}{rcll}
((\lambda x_2.M)\ ((\lambda x_1.M_2)\ M_1)) & \longrightarrow & ((\lambda x_1.((\lambda x_2.M)\ M_2))\ M_1) & (Comp) \\
((M\ N)\ L) & \longrightarrow & ((\lambda x.x\ L)\ (M\ N)) & (let.1) \\
(V\ (M\ N)) & \longrightarrow & ((\lambda x.V\ x)\ (M\ N)) & (let.2)
\end{array}$$

Figure 3: Additional Reductions for the Computational λ-calculus

We prove the equivalence of our calculus and $\boldsymbol{\lambda}_c$ by showing how each calculus proves the reductions of the other. The proof is tedious but straightforward.

Based on the above argument, the equivalence of the calculi yields the following reformulation of the Correspondence theorem.

Theorem 7.9′ (Correspondence (Reformulation)) *The calculi* $\boldsymbol{\lambda}\beta\eta$ *and* $\boldsymbol{\lambda}_c$ *are equivalent in the sense of Theorem 7.9.*

8 Observational Equivalence

The interest in calculi is motivated by their soundness with respect to observational equivalence (see Section 2). Therefore, the natural question is whether our extension is sound with respect to the call-by-value observational equivalence relation. Moggi [21] proves the result for a typed setting.

In a dynamically typed language, the soundness of the $\boldsymbol{\lambda}_c$-calculus with respect to the call-by-value observational equivalence relation depends on the particular language extensions. For example, the axiom η_v is *unsound* in languages like Lisp or Scheme as argued in Section 2. It is still possible to prove the soundness of the $\boldsymbol{\lambda}_c$-calculus for pure dynamically typed languages, i.e., languages with no constants.

Theorem 8.1 *Let $M, N \in \Lambda$. If $\boldsymbol{\lambda}_v A \vdash M = N$ then $M \cong_v N$.*

Proof. Let C be a context such that $C[M], C[N] \in \Lambda^0$. Assume $\boldsymbol{\lambda}_v A \vdash M = N$ and $eval_v(C[M])$ is defined. The goal is to show that $eval_v(C[N])$ is also defined.

It follows from the assumptions that $\boldsymbol{\lambda}_v A \vdash C[M] = C[N]$ by *Compatiblity*. Therefore,

$$\boldsymbol{\lambda}\beta\eta \ \vdash \ \mathcal{C}_k[\![C[M]]\!] = \mathcal{C}_k[\![C[N]]\!] \tag{1}$$

by Theorem 7.9. Also by the assumptions and the definition of $eval_v$, $\boldsymbol{\lambda}_v \vdash C[M] = V$ for some value V. Hence,

$$\boldsymbol{\lambda}\beta\eta \ \vdash \ \mathcal{C}_k[\![C[M]]\!] = \mathcal{C}_k[\![V]\!] \tag{2}$$

by Theorem 7.9. From (1) and (2), we deduce that $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![C[N]]\!] = \mathcal{C}_k[\![V]\!] = (k\ \Phi[\![V]\!])$. The Church-Rosser Theorem implies the existence of a term P such that:

$$\begin{aligned} \boldsymbol{\lambda}\beta\eta \ &\vdash \ \mathcal{C}_k[\![C[N]]\!] \twoheadrightarrow P \\ \boldsymbol{\lambda}\beta\eta \ &\vdash \ (k\ \Phi[\![V]\!]) \twoheadrightarrow P \end{aligned}$$

Obviously, all the reductions starting from the term $(k\ \Phi[\![V]\!])$ must occur inside $\Phi[\![V]\!]$. Since reductions preserve the syntactic categories in the CPS language, P must be of the form $(k\ W)$ for some W. Therefore, $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![C[N]]\!] \twoheadrightarrow (k\ W)$, and hence

$$\boldsymbol{\lambda}\beta\eta \ \vdash \ \mathcal{C}_k[\![C[N]]\!][k := \lambda x.x] \twoheadrightarrow ((\lambda x.x)\ W) \longrightarrow W$$

Lemma 8.2 implies that $eval_n((\mathcal{F}[\![C[N]]\!]\ \lambda x.x))$ is defined. Thus, $eval_v(C[N])$ is defined by Theorem 3.2. ∎

Lemma 8.2 *Let $M \in \Lambda^0$. If $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!][k := \lambda x.x] \twoheadrightarrow W$, then $eval_n(\mathcal{F}[\![M]\!]\ \lambda x.x)$ is defined.*

Proof. Assume $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![M]\!][k := \lambda x.x] \twoheadrightarrow W$, then by Lemma 5.6,

$$\boldsymbol{\lambda}\beta\eta \vdash (\mathcal{F}[\![M]\!]\ \lambda x.x) \twoheadrightarrow ((\lambda k.\mathcal{C}_k[\![M]\!])\ \lambda x.x) \longrightarrow \mathcal{C}_k[\![M]\!][k := \lambda x.x] \twoheadrightarrow W$$

By the Postponement Lemma [2: 15], we also have:

$$(\mathcal{F}[\![M]\!]\ \lambda x.x) \twoheadrightarrow_\beta L \twoheadrightarrow_\eta W \qquad \text{for some term } L$$

Since M is a closed term, W cannot be a variable; it must a λ-abstraction. Any *η-expansion* starting from a λ-abstraction will also result in a λ-abstraction. Therefore, L is a value:

$$(\mathcal{F}[\![M]\!]\ \lambda x.x) \twoheadrightarrow_\beta W'$$

By the Standard Reduction theorem [22], if a term reduces to a value, then it standard-reduces to a value. Therefore, $eval_n(\mathcal{F}[\![M]\!]\ \lambda x.x)$ is defined. ∎

As pointed out in Section 4, $M \cong_v N$ does *not* imply that $\mathcal{C}_k[\![M]\!] \cong_n \mathcal{C}_k[\![N]\!]$. For example, if

$$M \stackrel{df}{=} \lambda y.\lambda x.x\ (y\ x)$$
$$N \stackrel{df}{=} \lambda y.\lambda x.x\ (y\ \lambda z.xz)$$

then, $M \cong_v N$ [22]. On the other hand,

$$\begin{aligned}\mathcal{C}_k[\![M]\!] &= (k\ \lambda k.\lambda y.(k\ \lambda k.\lambda x.((y\ (x\ k))\ x))),\\ \mathcal{C}_k[\![N]\!] &= (k\ \lambda k.\lambda y.(k\ \lambda k.\lambda x.((y\ (x\ k))\ (\lambda k.\lambda z.((x\ k)\ z))))),\end{aligned}$$

and the context

$$\begin{aligned}D \stackrel{df}{=} &((\lambda k.[\,])\\ &\quad(\lambda a.((a\ (\lambda b.((b\ (\lambda x.x))\ \lambda d.\Omega)))\\ &\qquad(\lambda k.\lambda m.m\ (\lambda x.x)))))\end{aligned}$$

differentiates the two expressions. Since the context D includes a term $(\lambda k.\lambda m.(m\ \lambda x.x))$ that ignores its continuation, there is *no* context $C \in \Lambda$ such that $\mathcal{C}_k[\![C[M]]\!] = D[\mathcal{C}_k[\![M]\!]]$.

This result prompted Meyer and Riecke [19] to deduce that "continuations may be unreasonable". However, a restriction of D to range over contexts in the language $cps(\Lambda)$ results in a notion of observational equivalence that coincides with the call-by-value observational equivalence.

Definition 8.3. ($\cong_{cps(\Lambda)}$: *CPS observational equivalence*) Two terms P, Q are observationally equivalent, $P \cong_{cps(\Lambda)} Q$, iff for all contexts D such that $((\lambda k.D[P])\ (\lambda x.x))$, $((\lambda k.D[Q])\ (\lambda x.x))$ are programs in CPS form, then either both of $eval_n((\lambda k.D[P])\ \lambda x.x)$ and $eval_n((\lambda k.D[Q])\ \lambda x.x)$ are defined or both are undefined. ∎

Theorem 8.4 *Let $M, N \in \Lambda$. Then, $M \cong_v N$ iff $\mathcal{C}_k[\![M]\!] \cong_{cps(\Lambda)} \mathcal{C}_k[\![N]\!]$*

Proof. Assume $\mathcal{C}_k[\![M]\!] \ncong_{cps(\Lambda)} \mathcal{C}_k[\![N]\!]$, then there exists a context D such that $D[\mathcal{C}_k[\![M]\!]]$, $D[\mathcal{C}_k[\![N]\!]] \in cps(\Lambda)$ and the evaluation of one of the programs terminates while the other diverges. Without loss of generality, assume $eval_n((\lambda k.D[\mathcal{C}_k[\![M]\!]])\ \lambda x.x)$ is defined and $eval_n((\lambda k.D[\mathcal{C}_k[\![N]\!]])\ \lambda x.x)$ is undefined. By the definition of $eval_n$, we get:

$$\boldsymbol{\lambda}\beta \vdash ((\lambda k.D[\mathcal{C}_k[\![M]\!]])\ \lambda x.x) = W$$
$$\boldsymbol{\lambda}\beta \nvdash ((\lambda k.D[\mathcal{C}_k[\![N]\!]])\ \lambda x.x) = W' \qquad \text{for any } W'$$

By Lemma 8.5, there exists a context $C \in \Lambda$ such that:

$$\boldsymbol{\lambda}\beta\eta \vdash ((\lambda k.\mathcal{C}_k[\![C[M]]\!])\ \lambda x.x) = W$$
$$\boldsymbol{\lambda}\beta\eta \nvdash ((\lambda k.\mathcal{C}_k[\![C[N]]\!])\ \lambda x.x) = W' \qquad \text{for any } W'$$

By Lemma 8.2 and Theorem 3.2, $eval_v(C[M])$ is defined and $eval_v(C[N])$ is undefined. Therefore, $M \ncong_v N$. The reverse implication is straightforward. ∎

Lemma 8.5 *Let $M, N \in \Lambda$, $D[\mathcal{C}_k[\![M]\!]], D[\mathcal{C}_k[\![N]\!]] \in cps(\Lambda)$. Then, there exists a context C such that $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![C[M]]\!] = D[\mathcal{C}_k[\![M]\!]]$ and $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{C}_k[\![C[N]]\!] = D[\mathcal{C}_k[\![N]\!]]$.*

Proof. Since $D[\mathcal{C}_k[\![M]\!]], D[\mathcal{C}_k[\![N]\!]] \in cps(\Lambda)$, they are valid arguments to $\mathcal{C}^{-1}$. It suffices to show that the function $\mathcal{C}^{-1}$ is a homomorphism when restricted to inputs of the form $\mathcal{C}_k[\![M]\!]$ for some $M \in \Lambda$. By cases:

1. $\mathcal{C}^{-1}[\![(K\ W)]\!] = \mathcal{K}^{-1}[\![K]\!][\Phi^{-1}[\![W]\!]]$.
2. $\Phi^{-1}[\![x]\!] = x$.
3. $\Phi^{-1}[\![(\lambda k.k)]\!] = \lambda x.x$.
4. $\Phi^{-1}[\![(\lambda k.WK)]\!] = \lambda x.\mathcal{C}^{-1}[\![(W\ K)\ x]\!]$.
5. $\Phi^{-1}[\![(\lambda k.\lambda x.P)]\!] = \lambda x.\mathcal{C}^{-1}[\![P]\!]$.
6. $\mathcal{K}^{-1}[\![k]\!] = [\ ]$.
7. $\mathcal{K}^{-1}[\![(x\ K)]\!] = \mathcal{K}^{-1}[\![K]\!][(x\ [\ ])]$.
8. $\mathcal{K}^{-1}[\![((\lambda k.K_1)\ K_2)]\!] = \mathcal{K}^{-1}[\![K_1[k := K_2]]\!]$.
9. $\mathcal{K}^{-1}[\![(\lambda x.P)]\!] = ((\lambda x.\mathcal{C}^{-1}[\![P]\!])\ [\ ])$.

The function is homomorphic is all cases except cases 4 and 8. Both are impossible if the input is of the form $\mathcal{C}_k[\![M]\!]$ for some $M \in \Lambda$. ∎

9 Typed CPS Models

Friedman [12] investigates the connection between the typed $\beta\eta$-calculus and the functionals of finite type. Two convertible λ-terms define the same functional of finite type, and two non-convertible λ-terms define different functionals. Technically, $\boldsymbol{\lambda}\beta\eta$ is sound and complete with respect to the full type structure.

The Correspondence Theorem suggests the existence of a similar completeness theorem for the $\boldsymbol{\lambda}_c$-calculus and the CPS type structure [20, 14]. We prove this theorem after briefly reviewing the full type structure and the CPS type structure as denotational models for the call-by-value λ-calculus.

Full Type Structure

The set of types for the simply typed λ-calculus is

$$t ::= o \mid t \rightarrow t$$

where o denotes the *observable* base type. The denotations of types are as follows. Let B be an infinite set of elements of type o. Then, the full type pre-structure over B consists of a nonempty set D^t for each type t:

$$\begin{aligned} D^o &= B \\ D^{t_1 \rightarrow t_2} &= D^{t_1} \Rightarrow D^{t_2} \end{aligned}$$

The term language of the simply typed λ-calculus is the subset of Λ to which we can assign simple types as follows. Every variable has a fixed type t. If M is of type $t \rightarrow s$ and N is of type t, then $(M\ N)$ is of type s. Finally, if x is of type t and M is of type

s, then $\lambda x.M$ has type $t \to s$. The full type structure, $\mathcal{P}_B$, over B consists of a full type pre-structure and a *meaning* function $\mathcal{P}$ such that:

$$\begin{array}{rcll} Env : Vars & \Rightarrow & \cup_t D^t & \text{such that for } \rho \in Env,\ \rho(x^t) \in D^t \\ \mathcal{P}[\![x]\!]\rho & = & \rho(x) & \\ \mathcal{P}[\![\lambda x.M]\!]\rho & = & \underline{\lambda} a.\mathcal{P}[\![M]\!]\rho[x/a] & \\ \mathcal{P}[\![MN]\!]\rho & = & \mathcal{P}[\![M]\!]\rho(\mathcal{P}[\![N]\!]\rho) & \end{array}$$

The notation $A \Rightarrow B$ represents the set of functions from A to B and $\underline{\lambda} a.\cdots$ denotes the function f such that $f(a) = \cdots$. Also, $\mathcal{P}_B \models M = N$ means that for all environments ρ, $\mathcal{P}[\![M]\!]\rho = \mathcal{P}[\![N]\!]\rho$. Based on this definition, we can formulate the completeness theorem.

Theorem 9.1 (Friedman [12]) *Let $M, N \in \Lambda$. Then, $\boldsymbol{\lambda}\beta\eta \vdash M = N$ iff $\mathcal{P}_B \models M = N$.*

Since $\mathcal{F}[\![M]\!] \in \Lambda$, the theorem implies the following corollary.

Corollary 9.2 *Let $M, N \in \Lambda$. Then $\boldsymbol{\lambda}_c \vdash M = N$ iff $\boldsymbol{\lambda}\beta\eta \vdash \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$ iff $\mathcal{P}_B \models \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$.*

CPS Type Structure

Meyer and Wand [20] establish that the type of a Λ term may be related in a straightforward manner to the type of its CPS transform. The set of types becomes:[6]

$$s ::= o \mid (s \to a) \to (s \to a)$$

where a denotes a distinguished type of *answers*. The CPS type structure, $\mathcal{S}_B$, over some infinite base set B and an infinite set of answers A, consists of a nonempty set D^s for each type and a *meaning* function such that:

$$\begin{array}{rcl} D^o & = & B \\ D^{(s_1 \to a) \to s_2 \to a} & = & (D^{s_1} \Rightarrow A) \Rightarrow D^{s_2} \Rightarrow A \end{array}$$

and

$$\begin{array}{rcll} \mathcal{S} : \Lambda \times Env \times Continuation & \Rightarrow & A & \\ Env : Vars & \Rightarrow & \cup_s D^s & \text{such that for } \rho \in Env,\ \rho(x^s) \in D^s \\ Continuation : \cup_s D^s & \Rightarrow & A & \\ \mathcal{S}[\![x]\!]\rho\kappa & = & \kappa(\rho(x)) & \\ \mathcal{S}[\![\lambda x.M]\!]\rho\kappa & = & \kappa\ \underline{\lambda} c.\underline{\lambda} a.\mathcal{S}[\![M]\!]\rho[x/a]c & \\ \mathcal{S}[\![MN]\!]\rho\kappa & = & \mathcal{S}[\![M]\!]\rho(\underline{\lambda} a.\mathcal{S}[\![N]\!]\rho(\underline{\lambda} b.((a\ \kappa)\ b))) & \end{array}$$

Not surprisingly, the meaning of a term M in the CPS model is directly related to the meaning of $\mathcal{F}[\![M]\!]$ in the direct model.

[6] Meyer and Wand assume that the continuation is the second argument to a procedure. This means that their set of types is actually:

$$s ::= o \mid s \to (s \to a) \to a.$$

Lemma 9.3 *Let* $M \in \Lambda$. $(\mathcal{P}[\![\mathcal{F}[\![M]\!]]\!]\rho\ \kappa) = \mathcal{S}[\![M]\!]\rho\kappa$.

Proof Idea. The proof is by induction on M. ∎

This lemma implies that $\mathcal{S}_B$ and $\mathcal{P}_B$ satisfy theorems that are related via the Fischer-cps transformation.

Lemma 9.4 *Let* $M, N \in \Lambda$. *Then,* $\mathcal{P}_B \models \mathcal{F}[\![M]\!] = \mathcal{F}[\![N]\!]$ *iff* $\mathcal{S}_B \models M = N$.

Proof. The proposition follows from Lemma 9.3. ∎

It follows that the $\boldsymbol{\lambda}_c$-calculus is sound and complete with respect to the CPS type structure.

Theorem 9.5 *Let* $M, N \in \Lambda$. *Then,* $\boldsymbol{\lambda}_c \vdash M = N$ *iff* $\mathcal{S}_B \models M = N$.

Proof. The theorem is a direct consequence of Lemma 9.4 and Corollary 9.2. ∎

10 Conclusion and Future Research

In summary, our extensions of the λ_v-calculus result in an equational theory over Λ that is sound with respect to the call-by-value observational equivalence, and corresponds to $\boldsymbol{\lambda}\beta\eta$ over CPS terms. Thus far, we have also determined that the result extends to languages with ground constants and primitive functions and languages with imperative assignment procedures for data structures.

For languages with Scheme-like control operators, our extension of λ_v-calculus is still sound with respect to observational equivalence. However, the correspondence theorem fails since operators like *call/cc* manipulate their continuation in non-standard ways. To re-establish the correspondence theorem for such languages, we need to find an extension for the λ-control calculus [8, 9] that corresponds to $\boldsymbol{\lambda}\beta\eta$ on CPS terms.

References

1. Appel, A. and T. Jim. Continuation-passing, closure-passing style. In *Proc. 16th ACM Symposium on Principles of Programming Languages*, 1982, 293–302.
2. Barendregt, H.P. *The Lambda Calculus: Its Syntax and Semantics.* Revised Edition. Studies in Logic and the Foundations of Mathematics 103. North-Holland, Amsterdam, 1984.
3. Danvy, O. Back to direct style. In *4th Proc. European Symposium on Programming.* Springer Lecture Notes in Computer Science, 582. Springer Verlag, Berlin, 1992, 130–150.
4. Danvy, O. Three steps for the CPS transformation. Tech. Rep. CIS-92-2. Kansas State University, 1992.
5. Danvy, O. and A. Filinski. Representing control: A study of the CPS transformation. Tech. Rpt. CIS-91-2. Kansas State University, 1991.

6. Danvy, O. and J. L. Lawall. Back to direct style II: First-class continuations. In *Proc. 1992 ACM Conference on Lisp and Functional Programming*, 1992, this volume.

7. Felleisen, M. and D.P. Friedman. Control operators, the SECD-machine, and the λ-calculus. In *Formal Description of Programming Concepts III*, edited by M. Wirsing. Elsevier Science Publishers B.V. (North-Holland), Amsterdam, 1986, 193–217.

8. Felleisen, M. and R. Hieb. The revised report on the syntactic theories of sequential control and state. Technical Report 100, Rice University, June 1989. *Theor. Comput. Sci.*, 1992, to appear.

9. Felleisen, M., D.P. Friedman, E. Kohlbecker, and B. Duba. A syntactic theory of sequential control. *Theor. Comput. Sci.* **52**(3), 1987, 205–237. Preliminary version in: *Proc. Symposium on Logic in Computer Science*, 1986, 131–141.

10. Fischer, M.J. Lambda calculus schemata. In *Proc. ACM Conference on Proving Assertions About Programs, SIGPLAN Notices* **7**(1), 1972, 104–109.

11. Friedman, D.P., M. Wand, and C.T. Haynes. *Essentials of Programming Languages*. The MIT Press, Cambridge, Mass., 1992.

12. Friedman, H. Equality between functionals. In *Logic Colloquium'73*, Rohit Parikh (Ed.), Lecture Notes in Mathematics 453, Springer Verlag, Berlin, 1973, 22–37.

13. Gateley, J. and B.F. Duba. Call-by-value combinatory logic and the lambda-value calculus. In *Proc. 1991 Workshop on Mathematical Foundations of Programming Semantics*. Lecture Notes in Computer Science 517, to appear.

14. Harper, R. and M. Lillibridge. Polymorphic type assignment and cps conversion. Tech. Rpt. CMU-CS-92-122. Carnegie Mellon University. 1992. Preliminary version in: *Proc. Workshop on Continuations: CW92*. Technical Report CS-92-1426. Stanford University, 1992, 13–22.

15. Hieb R., R. K. Dybvig, and C. Bruggeman. Representing control in the presence of first-class continuations. In *Proceedings of the SIGPLAN '90 Conference on Programming Language Design and Implementation*, June 1990, 66–77.

16. Kranz, D., et al. ORBIT: An optimizing compiler for Scheme. In *Proc. SIGPLAN 1986 Symposium on Compiler Construction. SIGPLAN Notices* **21**(7), 1986, 219–233.

17. Landin, P.J. The mechanical evaluation of expressions. *Comput. J.* **6**(4), 1964, 308–320.

18. Leroy, X. The Zinc experiment. Technical Report 117. INRIA, 1990.

19. Meyer, A.R. and J.R. Riecke. Continuations may be unreasonable. In *Proc. 1988 Conference on Lisp and Functional Programming*, 1988, 63–71.

20. Meyer, A.R. and M. Wand. Continuation semantics in typed lambda-calculi. *Proc. Workshop Logics of Programs*, Lecture Notes in Computer Science 193, Springer-Verlag, Heidelberg, 1985, 219–224.

21. Moggi, E. Computational lambda-calculus and monads. In *Proc. Symposium on Logic in Computer Science*, 1989, 14–23. Also appeared as: LFCS Report ECS-LFCS-88-66, University of Edinburgh, 1988.

22. Plotkin, G.D. Call-by-name, call-by-value, and the λ-calculus. *Theor. Comput. Sci.* **1**, 1975, 125–159.

23. Reynolds, J.C. Definitional interpreters for higher-order programming languages. In *Proc. ACM Annual Conference*, 1972, 717–740.

24. Shivers, O. Control-flow Analysis of Higher-Order Languages or Taming Lambda. Ph.D. dissertation, Carnegie-Mellon University, 1991.

25. Steele, G.L., Jr. RABBIT: A compiler for SCHEME. Memo 474, MIT AI Lab, 1978.