A Sound and Complete Axiomatization of Delimited Continuations

Yukiyoshi Kameyama
Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Japan
and Japan Science and Technology Corporation
kam@is.tsukuba.ac.jp

Masahito Hasegawa
Research Institute for Mathematical Sciences
Kyoto University
Kyoto, Japan
and Japan Science and Technology Corporation
hassei@kurims.kyoto-u.ac.jp

ABSTRACT

The shift and reset operators, proposed by Danvy and Filinski, are powerful control primitives for capturing *delimited continuations*. Delimited continuation is a similar concept as the standard (unlimited) continuation, but it represents part of the rest of the computation, rather than the whole rest of computation. In the literature, the semantics of shift and reset has been given by a CPS-translation only. This paper gives a direct axiomatization of calculus with shift and reset, namely, we introduce a set of equations, and prove that it is sound and complete with respect to the CPS-translation. We also introduce a calculus with control operators which is as expressive as the calculus with shift and reset, has a sound and complete axiomatization, and is conservative over Sabry and Felleisen's theory for first-class continuations.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages

General Terms

Languages,Theory,Verification

Keywords

Continuation, CPS-translation, Axiomatization

1. INTRODUCTION

First-class continuations are powerful control facility in functional programming languages such as Scheme (`call/cc`) and SML/NJ (`callcc` and `throw`). It captures the whole rest of computation so that we can represent loops, backtracking, coroutines, and other control structures [29]. However, one sometimes wants to capture some part of the rest of computation, and compose the captured continuation with ordinary functions. In the literature, such a kind of continuations is called *partial* continuations, *delimited* continuations, *functional* continuations, or *composable* continuations, and we use the term "delimited continuations" in this paper. Recent years have found that delimited continuations are useful in writing programs with explicit control in such areas as partial evaluation, CPS-translation, and even mobile computation [32, 30, 27, 2].

There are many proposals for representing first-class delimited continuations [9, 6, 23, 17, 14]. Among them, Danvy and Filinski's shift and reset operators [6, 7] are most widely used and efficient implementations for them have been proposed [11, 13]. An important merit of their operators is that a clean and rigorous semantics is given through a CPS-translation. Moreover, Filinski [11] proved that *any* expressible[1] monadic effects are representable by the shift and reset operators, which demonstrates the expressive power of these operators.

We address the problem of reasoning about programs with first-class delimited continuations, in particular the shift and reset operators. Although there is a rigorous CPS-based semantics for shift and reset, the CPS-translation generates larger and restructured codes, with which it is often very hard to recognize the high-level aspects of the original program (though the size of the generated codes is not really a matter, if we use compacting CPS-translations [25, 8]). Thus we look for a direct axiomatization which is closer to the programmer's intuition and yet as powerful as the CPS-based reasoning.

In this paper, we give such an axiomatization for shift and reset, and prove the soundness and completeness of our axiomatization with respect to the semantics given by CPS-translation. Based on our axiomatization, one can directly reason about properties of programs which involve shift and reset, without converting the program to CPS-terms.

Our method is an extension of the technique developed by Sabry and Felleisen [25], who gave a sound and complete axiomatization of call-by-value type-free lambda calculus with first-class continuations. There are mainly two obstacles when we apply their technique to shift and reset.

The first one is that, the target calculus of the CPS-translation in the case of shift and reset is not the usual

[1]He defines that a monad is expressible if its defining functions η and $\_^*$ are defined in purely functional object language.

ICFP'03, August 25–29, 2003, Uppsala, Sweden.

CPS-forms, by which we mean there are non-value terms in an argument position in functional application. This means that we cannot use the full $\beta\eta$-equality in the target calculus, which is extensively used in their proof. To avoid this problem we use Danvy and Filinski's CPS-hierarchy, which can be considered as doubling CPS-translation, that is, to CPS-translate source terms twice (though this is only conceptual. In this paper we use a curried version of CPS-hierarchy, which does not exactly agree with the composition of two CPS-translations.)

The second problem is the complexity of this doubling CPS-translation. In the target calculus of CPS-translation, we must treat not only an ordinary continuation variable, but also a metacontinuation variable which is introduced by the second CPS-translation, and the term structure of the target calculus is more complex than Sabry-Felleisen's. Also, we have two operators shift and reset, rather than a single primitive callcc, which also complicates the proof. Nevertheless, by making clear distinction between pure evaluation contexts and ordinary evaluation contexts, and carefully treating the reset primitive, we are able to extend Sabry and Felleisen's technique to the shift and reset case, and find a sound and complete axiomatization.

Independently to Sabry and Felleisen, Hofmann [18] also proposed a sound and complete axiomatization for a call-by-value lambda calculus with first-class continuations. He constructed a term model using a category-theoretic machinery to prove the result, which is more structured than Sabry and Felleisen's. However, he treated a typed version, and it is not certain that his proof technique can be applied to the type-free case. Recently Führmann and Thielecke presented a refined proof and axiomatization in a typed setting [12].

Sabry [24] proposed an elegant technique to obtain a sound and complete axiomatization with respect to a CPS-semantics. He applied it to a calculus with shift and *lazy* reset, which differs from (ordinary) reset in that a reset term is always a value. As a by-product of his proof, he showed some axioms by Sabry and Felleisen [25] are redundant. In order to apply Sabry's technique to the calculus for shift and reset, we must carefully reformulate the target calculus since the target calculus in our case does not have the full $\beta\eta$-equality while we need the $\beta\eta$-equality for continuations (and metacontinuations). Yet, we must note that the resulting axioms in this work are very close to Sabry's, and in fact, his axioms can be obtained by adding one axiom to ours. We shall use this fact to obtain conservativity of our axioms over Sabry-Felleisen's axiomatization.

The rest of this paper is organized as follows. We introduce the shift and reset operators and their CPS-translation in Section 2. Our axiomatization with the soundness theorem is given in Section 3. Section 4 analyzes the target calculus and gives an inverse translation from the target to the source. Section 5 and Section 6 give a proof of the completeness of our axioms. Section 7 examines the axioms given in this paper, specifically, gives conservativity and some independence results. Section 8 gives concluding remarks.

2. SHIFT AND RESET

2.1 Source Language

In this section we introduce the shift and reset operators due to Danvy and Filinski [6, 7] in a type-free, call-by-value setting.

The grammar of the calculus $\lambda_{\mathcal{S}}$ is given by:

$$\begin{array}{rrcl} \text{(terms)} & M, N & ::= & V \mid MN \mid \langle M \rangle \\ \text{(values)} & V & ::= & x \mid \lambda x.M \mid \mathcal{S} \end{array}$$

where x is a variable, MN and $\lambda x.M$ are application and λ-abstraction.

The shift operator resembles the standard first-class continuation operator callcc, but it captures part of the rest of the computation, and the part is specified by the (dynamically determined) closest reset operator. In the calculus $\lambda_{\mathcal{S}}$, the angle brackets in the term $\langle M \rangle$ represent the reset operator, namely, they delimit the effect of the shift operators in M. We give the shift operator as a constant, rather than the original form $\xi c.M$, which can be defined by $\mathcal{S}(\lambda c.M)$ in our formulation.

Free and bound occurrences of variables are defined as usual. $FV(M)$ denotes the set of free variables in M. The usual capture-avoiding substitution is denoted by $M\{x := N\}$.

We introduce three kinds of contexts.

$$\begin{array}{rrcl} \text{(contexts)} & C & ::= & [\,] \mid \lambda x.C \mid CM \mid MC \mid \langle C \rangle \\ \text{(e-contexts)} & E & ::= & [\,] \mid EM \mid VE \mid \langle E \rangle \\ \text{(p-contexts)} & F & ::= & [\,] \mid FM \mid VF \end{array}$$

where C stands for contexts, E for evaluation contexts (e-contexts, for short), and F for pure evaluation contexts (p-contexts, for short). Contexts and evaluation contexts are standard [10]. We distinguish from other contexts a pure evaluation context, which is an evaluation context that does not have any enclosing reset operators around the hole. We use the metavariables E and F (with suffixes) for e-contexts and p-contexts, resp. For a context C and a term M, we write $C[M]$ for the usual (variable-capturing) hole-filling operation. For instance, if $C \equiv \lambda x.[\,]$ and $M \equiv xy$, then $C[M] \equiv \lambda x.xy$, thus x is captured by C. If a variable x becomes a bound variable in $C[x]$, we say it is *captured* by C.

An intuitive operational semantics of the shift and reset operators can be given as follows. Let E, F and V be an e-context, a p-context and a value, resp. We have the following equalities (or reductions if we read from left to right):

$$\begin{array}{rcl} E[\langle F[\mathcal{S}V] \rangle] & = & E[\langle V(\lambda x.\langle F[x] \rangle) \rangle] \\ E[\langle V \rangle] & = & E[V] \end{array}$$

In the first line, the shift operator $\mathcal{S}$ encapsulates part of the rest of the computation as a function $\lambda x.\langle F[x] \rangle$. The context being captured is specified by the closest reset operator, since F is a p-context, and there are no other reset's enclosing the hole. The captured computation is almost $\lambda x.F[x]$, but it contains an additional reset operator, which correctly delimits the (possible) effects in F; this also allows a simple CPS-translation, as explained by Danvy and Filinski [6]. Note that, the current delimited continuation F is discarded, so the shift operator also behaves like an abort. In the second line, the reset operator is discarded if its content is already a value.

In this paper we also consider other control operators $\mathcal{D}$ and $\mathcal{A}$, whose intuitive semantics is given as follows:

$$E[\langle F[\mathcal{D}V]\rangle] = E[\langle F[V(\lambda x.\mathcal{A}(F[x]))]\rangle]$$
$$E[\langle F[\mathcal{A}V]\rangle] = E[V]$$

Since the shift operator does two things (capturing a delimited continuation and aborting), we want to split its roles into two operators to help our technical development. The abortive behavior of shift is inherited by the $\mathcal{A}$-operator, and the role of capturing a delimited continuation is inherited by the $\mathcal{D}$-operator. By replacing the shift operator by the $\mathcal{D}$- and $\mathcal{A}$-operators we obtain the calculus $\lambda_{\mathcal{DA}}$.

Note that $\mathcal{S}$ and $\mathcal{D}, \mathcal{A}$ are inter-definable by each other. That is, $\lambda_{\mathcal{S}}$ can define $\lambda_{\mathcal{DA}}$, and vice versa.

$$\begin{aligned}\mathcal{D} &= \lambda z.\mathcal{S}(\lambda k.k(z(\lambda x.\mathcal{A}(kx))))\\ \mathcal{A} &= \lambda z.\mathcal{S}(\lambda d.z)\\ \mathcal{S} &= \lambda z.\mathcal{D}(\lambda k.\mathcal{A}(z(\lambda x.\langle kx\rangle)))\end{aligned}$$

Formally, these equations are justified by the CPS-translation given later.

The decomposition of the shift operator to the $\mathcal{D}$- and $\mathcal{A}$-operators are not new to this work, and we refer to Murthy [21] and Filinski [11] for interested readers.

In this paper, we shall first axiomatize the calculus $\lambda_{\mathcal{DA}}$, then the calculus $\lambda_{\mathcal{S}}$.

2.2 CPS-translation

A CPS-translation is a syntactic translation from source terms to target terms. In our case, the source calculus is $\lambda_{\mathcal{S}}$ and $\lambda_{\mathcal{DA}}$, and the target language is the pure type-free lambda calculus without control operators.

As explained in the introduction, we shall use Danvy and Filinski's CPS-hierarchy (metacontinuation semantics). The reason is as follows: the image of the ordinary CPS-translation of a shift-term contains a term like $k'(kx)$, whose evaluation is sensitive to the evaluation order. Since we want to evaluate it in call-by-value, the full β-reduction cannot be used in the target language. On the other hand, if we adopt the CPS-hierarchy (twice or more CPS-translations), target terms do not contain such forms, and we can safely use the full $\beta\eta$-equality in the target calculus.

Before giving the CPS-translation we introduce a combinator J in the target calculus defined by $J = \lambda x\gamma.\gamma x$. It is the image of the empty context (the identity continuation).

The CPS-translations of $\lambda_{\mathcal{S}}$ and $\lambda_{\mathcal{DA}}$ in Plotkin style are given as follows, where the introduced bound variables k, k', x, v, w, m, n, and γ are assumed to be fresh.

$$\begin{aligned}[\_] &: \text{Term} \to \text{Target Term}\\ [V] &= \lambda k.kV^*\\ [MN] &= \lambda k.[M](\lambda m.[N](\lambda n.mnk))\\ [\langle M\rangle] &= \lambda k\gamma.[M]J(\lambda v.kv\gamma)\end{aligned}$$

$$\begin{aligned}(\_)^* &: \text{Value} \to \text{Target Term}\\ x^* &= x\\ (\lambda x.M)^* &= \lambda x.[M]\\ \mathcal{S}^* &= \lambda xk.x(\lambda vk'\gamma.kv(\lambda w.k'w\gamma))J\\ \mathcal{D}^* &= \lambda xk.x(\lambda vk'.kv)k\\ \mathcal{A}^* &= \lambda xk\gamma.\gamma x\end{aligned}$$

Note that the result of translating the term $\langle M\rangle$ needs two arguments k and γ. The first one is the ordinary continuation parameter, while the second one is the metacontinuation parameter.

An important observation on the CPS-translation above is that, while the continuation variable k can be used many times (by the $\mathcal{D}$-operator), the metacontinuation variable γ is always used once, thus *linear*. Intuitively, it can be explained like this: the CPS-translation is essentially the composition of two successive (ordinary) CPS-translations, and the source language of the second CPS-translation is the ordinary call-by-value lambda calculus without control operators. Therefore, as shown in the literature [5, 4, 15], continuations are used linearly and the target calculus of the second CPS-translation can be considered as linear lambda calculus. Continuations in the second CPS-translation are metacontinuations in our setting, thus γ is linear in the target calculus. The second author exploits this linearity in his semantic analysis of delimited continuations [16].

Although we present the CPS-translation in a type-free setting, we may consider its target calculus as typed calculus with recursive types, and the mentioned linearity is made explicit in the type system. We will briefly mention in the conclusion how it is done.

The CPS-translation above defines a rigorous semantics of shift and reset. The reset operator installs the identity continuation J as the current continuation. The $\mathcal{D}$-operator captures the current continuation k (up to the most recently called reset operator), and applies $\lambda vk'.kv$ to the argument. The $\mathcal{A}$-operator discards the current continuation k, and directly applies the argument x to the metacontinuation γ.

3. AXIOMS

For the purpose of presentation, we give the axioms in this section, and postpone the proof of the completeness to later sections. The actual development was in the opposite order; these axioms have been obtained during the proof of completeness.

3.1 Axioms and their Soundness

The set of axioms for $\lambda_{\mathcal{DA}}$ is given in Figure 1, and that for $\lambda_{\mathcal{S}}$ is given Figure 2.

For the sake of comparison, we list Sabry and Felleisen's axiomatization for first-class continuations [25] in Figure 3. (we slightly changed the names of the axioms from the original.)

Some explanations of our axioms will follow.

The axioms β_v, η_v and β_Ω are axioms for pure lambda terms, and are essentially the same as those for Moggi's computational lambda calculus [20], which is considered as the canonical calculus in call-by-value. Sabry and Felleisen's axiomatizations contains the axiom β-lift, but it is known to be derivable[2].

The axioms $\mathcal{D}$-elim, $\mathcal{D}$-current, $\mathcal{D}$-abort, and $\mathcal{D}$-lift are axioms for the $\mathcal{D}$-operator. It is easy to see that these axioms in our axiomatization (Figure 1) are essentially equivalent to Sabry and Felleisen's axiomatization (Figure 3) for unlimited continuations if we identify $\mathcal{D}$ with callcc. The only difference is that the latter contains an extra axiom (callcc-tail), but it is derivable as is shown below.

The remaining axioms are the key to the axiomatization

[2]Sabry [24] attributes this observation to Filinski.

$$
\begin{array}{rll}
(\lambda x.M)V = M\{x := V\} & & \beta_v \\
\lambda x.\ Vx = V & & \eta_v, \text{ if } x \notin FV(V) \\
(\lambda x.F[x])M = F[M] & & \beta_\Omega, \text{ if } x \notin FV(F) \\
\langle V \rangle = V & & \text{reset-value} \\
\langle (\lambda x.M)\langle N \rangle \rangle = (\lambda x.\langle M \rangle)\langle N \rangle & & \text{reset-lift} \\
 & & \\
\mathcal{D}(\lambda k.M) = M & & \mathcal{D}\text{-elim, if } k \notin FV(M) \\
\mathcal{D}(\lambda k.kM) = \mathcal{D}(\lambda k.M) & & \mathcal{D}\text{-current} \\
\mathcal{D}(\lambda k.C[F[kM]]) = \mathcal{D}(\lambda k.C[kM]) & & \mathcal{D}\text{-abort, if } k \text{ is not captured by } C \\
F[\mathcal{D}M] = \mathcal{D}(\lambda k.F[M(\lambda f.kF[f])]) & & \mathcal{D}\text{-lift, if } k \notin FV(F[\mathcal{D}M]) \text{ and } f \notin FV(kF) \\
F[\mathcal{A}M] = \mathcal{A}M & & \mathcal{A}\text{-lift} \\
\langle \mathcal{A}M \rangle = \langle M \rangle & & \mathcal{A}\text{-top} \\
\mathcal{A}\langle M \rangle = \mathcal{A}M & & \mathcal{A}\text{-reset}
\end{array}
$$

Figure 1: Axioms for $\lambda_{\mathcal{DA}}$

Axioms β_v, η_v, β_Ω, reset-value, and reset-lift with the following axioms.

$$
\begin{array}{rll}
\mathcal{S}(\lambda k.kM) = M & & \mathcal{S}\text{-elim, if } k \notin FV(M) \\
\langle F[\mathcal{S}M] \rangle = \langle M(\lambda x.\langle F[x] \rangle) \rangle & & \text{reset-}\mathcal{S}\text{, if } x \notin FV(F) \\
\mathcal{S}(\lambda k.\langle M \rangle) = \mathcal{S}(\lambda k.M) & & \mathcal{S}\text{-reset}
\end{array}
$$

Figure 2: Axioms for $\lambda_{\mathcal{S}}$

of delimited continuations. The axioms $\mathcal{A}$-lift, $\mathcal{A}$-top, $\mathcal{A}$-reset, and reset-value are already present in Sabry's work [24], but the notion of values in his calculus is different from that in our calculus in that he treated "lazy" prompt (reset) and $\langle M \rangle$ is always a value.

The axiom reset-lift seems new to this work. It states a non-trivial equality on the nested and sequential uses of the reset operator, and is also the key case of the equation β_Ω-reset-2 in Lemma 1 below.

For $\lambda_{\mathcal{S}}$, we replace the seven axioms for $\mathcal{D}$ and $\mathcal{A}$ by three axioms for $\mathcal{S}$, which closely resemble those for Felleisen's $\mathcal{C}$-operator [10] except that we need some extra care for the interaction with the reset operator, and that the continuation captured by $\mathcal{S}$ is composable (not abortive).

Let $\mathcal{T}$ be one of the theories $\lambda_{\mathcal{DA}}$, $\lambda_{\mathcal{S}}$, and $\lambda_{\beta\eta}$, where $\lambda_{\beta\eta}$ is the pure lambda calculus with the full $\beta\eta$-equality. We write $\mathcal{T} \vdash M = N$ if $M = N$ is derived from the axioms in $\mathcal{T}$. We also write $M \equiv N$ for the syntactic equality (including α-equivalence).

THEOREM 1 (SOUNDNESS). *Let $\mathcal{T}$ be $\lambda_{\mathcal{DA}}$ or $\lambda_{\mathcal{S}}$, and M and N be terms in $\mathcal{T}$. Then $\mathcal{T} \vdash M = N$ implies $\lambda_{\beta\eta} \vdash [M] = [N]$.*

This theorem is proved by directly computing both sides of each axiom, and the proof is omitted.

3.2 Derivable Equations

The next two lemmas give several useful equations in $\lambda_{\mathcal{DA}}$ and $\lambda_{\mathcal{S}}$. The proofs of these lemmas can be found in the appendix.

LEMMA 1. *The following equations are derivable in $\lambda_{\mathcal{DA}}$, in which we assume $x \notin FV(F) \cup FV(E) \cup \{k\}$ and $k \notin FV(M_2)$.*

$$
\begin{array}{rr}
F[(\lambda x.N)M] = (\lambda x.F[N])M & \beta\textit{-lift} \\
\langle\langle M \rangle\rangle = \langle M \rangle & \textit{reset-reset} \\
\langle \mathcal{D}M \rangle = \langle M\mathcal{A} \rangle & \mathcal{D}\textit{-top} \\
(\lambda x.\mathcal{D}(\lambda k.M_1))M_2 = \mathcal{D}(\lambda k.(\lambda x.M_1)M_2) & \mathcal{D}\textit{-tail} \\
\langle (\lambda x.\langle F[x] \rangle)M \rangle = \langle F[M] \rangle & \beta_\Omega\textit{-reset-1} \\
(\lambda x.E[x])\langle M \rangle = E[\langle M \rangle] & \beta_\Omega\textit{-reset-2}
\end{array}
$$

The axiom β_Ω-reset-2 is similar to β_Ω, but the former allows an arbitrary e-context E in the body of the β-redex, rather than a p-context F (which must not have reset-operators enclosing the hole). Note that a more general equation $(\lambda x.E[x])M = E[M]$ is not sound with respect to the CPS-translation.

LEMMA 2. *The following equations are derivable in $\lambda_{\mathcal{S}}$, in which we assume $x \notin FV(kF)$ and $k \notin FV(F) \cup FV(M)$.*

$$
\begin{array}{rr}
F[\mathcal{S}M] = \mathcal{S}(\lambda k.M(\lambda x.\langle kF[x] \rangle))\mathcal{S}\textit{-nat} \\
\mathcal{S}(\lambda k.Mk) = \mathcal{S}M & \mathcal{S}_\eta \\
(\lambda x.\mathcal{S}(\lambda k.N))M = \mathcal{S}(\lambda k.(\lambda x.N)M) & \mathcal{S}\textit{-tail}
\end{array}
$$

Note that the equation $\mathcal{S}$-nat has a similar form to $\mathcal{D}$-lift, however, it can be derived in $\lambda_{\mathcal{S}}$.

4. TARGET CALCULUS AND INVERSE TRANSLATION

The goal of the rest of this paper is to prove the completeness of the axioms for $\lambda_{\mathcal{DA}}$ and $\lambda_{\mathcal{S}}$. There are three ways to prove the completeness of this kind. The first approach is due to Sabry and Felleisen [25], who carefully analyze the

Axioms β_v, η_v, β_Ω, and $\mathcal{A}$-lift with the following axioms.

$$
\begin{array}{rll}
F[(\lambda x.N)M] = (\lambda x.F[N])M & & \beta\text{-lift} \\
\mathsf{callcc}(\lambda k.M) = M & & \mathsf{callcc}\text{-elim, if } k \notin FV(M) \\
\mathsf{callcc}(\lambda k.kM) = \mathsf{callcc}(\lambda k.M) & & \mathsf{callcc}\text{-current} \\
\mathsf{callcc}(\lambda k.C[F[kM]]) = \mathsf{callcc}(\lambda k.C[kM]) & & \mathsf{callcc}\text{-abort, if } k \text{ is not captured by } C \\
F[\mathsf{callcc}M] = \mathsf{callcc}(\lambda k.F[M(\lambda f.kF[f])]) & & \mathsf{callcc}\text{-lift, if } k \notin FV(F[M]) \text{ and } f \notin FV(kF) \\
(\lambda x.\mathsf{callcc}(\lambda k.M_1))M_2 = \mathsf{callcc}(\lambda k.(\lambda x.M_1)M_2) & & \mathsf{callcc}\text{-tail}
\end{array}
$$

Figure 3: Sabry and Felleisen's Axioms for callcc

target of the CPS-translation, then define an "inverse" of the CPS-translation, and prove the equality in the target (the $\beta\eta$-equality) is preserved by the inverse translation. The other two approaches are Hofmann's work [18] and Sabry's work [24], both of which are cleaner than Sabry-Felleisen's proof. However, it is not certain whether we can apply these techniques to the calculus with shift and reset. Hofmann's work is for the typed-setting. It seems possible to apply Sabry's work to our case, but it needs reformulation of the target calculus. In order to axiomatize the very calculus Danvy and Filinski proposed, we will adopt Sabry-Felleisen's approach in this paper.

In this and the next sections, we take $\lambda_{\mathcal{DA}}$ as the source calculus; the completeness for $\lambda_{\mathcal{S}}$ is derived in Section 6 as a corollary to the case of $\lambda_{\mathcal{DA}}$.

4.1 The Target Calculus

We first analyze the structure of the target terms of the CPS-translation which is necessary to define an inverse of the CPS-translation.

Assume that the ordinary variables are $x_1, \cdots x_n$, the continuation variables are $k_1, \cdots, k_m$, and the metacontinuation variable is γ. By the linearity condition of the metacontinuation variable, we need only one metacontinuation variable.

The following grammar defines the target calculus.

$$
\begin{array}{rcl}
\text{(terms) } T & ::= & \lambda k.Q \mid WW \\
\text{(pre-values) } Q & ::= & \lambda\gamma.R \mid TK \mid KW \\
\text{(answers) } R & ::= & QG \mid GW \\
\text{(values) } W & ::= & x_1 \mid \cdots \mid x_n \mid \lambda x.T \\
\text{(continuations) } K & ::= & k_1 \mid \cdots \mid k_m \mid \lambda x.Q \\
\text{(metacontinuations) } G & ::= & \gamma \mid \lambda x.R
\end{array}
$$

A term in the target calculus is called a T-term if it belongs to the class T defined above. Similarly, a Q-, R-, W-, K-, and G-term are defined.

Since there is only one metacontinuation variable γ, the term $\lambda\gamma.R$ does not have free metacontinuation variables. By further extending this reasoning, we have that T-, Q-, W- and K-terms do not contain free metacontinuation variables, and R- and G-terms contain a single free occurrence of the metacontinuation variable γ.

THEOREM 2. *(i) Let M and V be a term and a value in $\lambda_{\mathcal{DA}}$, resp. Then $[M]$ is a T-term and V^* is a W-term in the target calculus.*

(ii) The target calculus is closed under the full $\beta\eta$-reductions.

PROOF. (i) We have that $\mathcal{D}^*$ and $\mathcal{A}^*$ are W-terms, and J is a K-term. Other cases are proved by straightforward induction.

(ii) This is easily proved by case-analysis. We must be careful to check that the linearity condition on the metacontinuation variable γ is preserved through the reduction, but it is proved by straightforward induction. □

Even though the above grammar is closed under the reductions in the target calculus, there are T-terms which are not an image of any source term. However, in the next subsection, we see that an "inverse" function from *all* the T-terms to the source terms can be defined.

4.2 Inverse Translation

The "inverse" of the CPS-translation is a total function from the set of target terms (not necessarily in the image of CPS-translation, but arbitrary terms defined by the above grammar) to the set of terms in $\lambda_{\mathcal{DA}}$. Depending on the six classes in the target calculus, the inverse translation consists of the six functions, $\mathcal{T}^{-1}(\_)$, $\mathcal{Q}^{-1}(\_)$, $\mathcal{R}^{-1}(\_)$, $\mathcal{W}^{-1}(\_)$, $\mathcal{K}^{-1}(\_)$, and $\mathcal{G}^{-1}(\_)$ defined below.

$$
\begin{array}{rcl}
\mathcal{T}^{-1}(\lambda k.Q) & = & \mathcal{D}(\lambda k.\mathcal{A}(\mathcal{Q}^{-1}(Q))) \\
\mathcal{T}^{-1}(W_1W_2) & = & \mathcal{W}^{-1}(W_1)\mathcal{W}^{-1}(W_2) \\
\mathcal{Q}^{-1}(\lambda\gamma.R) & = & \mathcal{D}(\lambda\gamma.\mathcal{R}^{-1}(R)) \\
\mathcal{Q}^{-1}(TK) & = & \mathcal{K}^{-1}(K)[\mathcal{T}^{-1}(T)] \\
\mathcal{Q}^{-1}(KW) & = & \mathcal{K}^{-1}(K)[\mathcal{W}^{-1}(W)] \\
\mathcal{R}^{-1}(QG) & = & \mathcal{G}^{-1}(G)[\langle\mathcal{Q}^{-1}(Q)\rangle] \\
\mathcal{R}^{-1}(GW) & = & \mathcal{G}^{-1}(G)[\mathcal{W}^{-1}(W)] \\
\mathcal{W}^{-1}(x) & = & x \\
\mathcal{W}^{-1}(\lambda x.T) & = & \lambda x.\mathcal{T}^{-1}(T) \\
\mathcal{K}^{-1}(k) & = & k[\ ] \\
\mathcal{K}^{-1}(\lambda x.Q) & = & (\lambda x.\langle\mathcal{Q}^{-1}(Q)\rangle)[\ ] \\
\mathcal{G}^{-1}(\gamma) & = & \gamma[\ ] \\
\mathcal{G}^{-1}(\lambda x.R) & = & (\lambda x.\mathcal{R}^{-1}(R))[\ ]
\end{array}
$$

The inverse translation is similar in spirit to Sabry and Felleisen's inverse translation [25]. For instance, we explicitly capture continuations in the cases of $\mathcal{T}^{-1}(\lambda k.Q)$ and $\mathcal{Q}^{-1}(\lambda\gamma.R)$. Yet, there are differences in treating values and continuations, and these are subtle points. For instance, a reset (delimiter) is inserted to the result of $\mathcal{K}^{-1}(\lambda x.Q)$, which is essential for our completeness proof.

As an example of the inverse translation, let us consider the term $\lambda k.fx(\lambda v.k(\lambda ak.ka))$ in the target calculus. This term is obtained by β-reducing $[(\lambda ya.a)(fx)]$. We can com-

pute its inverse as follows:

$$\begin{array}{rcl} & \mathcal{T}^{-1}(\lambda k.fx(\lambda v.k(\lambda ak'.k'a))) \\ \equiv & \mathcal{D}(\lambda k.\mathcal{A}(\mathcal{Q}^{-1}(fx(\lambda v.k(\lambda ak'.k'a))))) \\ \equiv & \mathcal{D}(\lambda k.\mathcal{A}(\mathcal{K}^{-1}(\lambda v.k(\lambda ak'.k'a))[\mathcal{T}^{-1}(fx)])) \end{array}$$

By a similar calculation, we have

$$\begin{array}{rcl} \mathcal{K}^{-1}(\lambda v.k(\lambda ak'.k'a)) & \equiv & (\lambda v.\langle k(\lambda a.\mathcal{D}(\lambda k'.\mathcal{A}(k'a)))\rangle)[\,], \\ \mathcal{T}^{-1}(fx) & \equiv & fx \end{array}$$

thus, the result of the inversion is

$$\mathcal{D}(\lambda k.\mathcal{A}((\lambda v.\langle k(\lambda a.\mathcal{D}(\lambda k'.\mathcal{A}(k'a)))\rangle)(fx))).$$

We can show the resulting term is equal to $(\lambda va.a)(fx)$ under the theory $\lambda_{\mathcal{DA}}$.

Going back to the definition of the inverse translation, let us observe that its images have the following special form:

1. $\mathcal{T}^{-1}(T)$ is a term, $\mathcal{W}^{-1}(W)$ is a value, and $\mathcal{K}^{-1}(K)$ and $\mathcal{G}^{-1}(G)$ are p-contexts in $\lambda_{\mathcal{DA}}$.
2. Whenever $\mathcal{Q}^{-1}(Q)$ appears as subterms of other terms, it is either of the form $\langle\mathcal{Q}^{-1}(Q)\rangle$ or the form $\mathcal{A}(\mathcal{Q}^{-1}(Q))$. Namely, $\mathcal{Q}^{-1}(Q)$ always appears in a delimited context, or an abortive context. The same thing holds for $\mathcal{K}^{-1}(K)$.
3. Whenever $\mathcal{G}^{-1}(G)[M]$ appears as a subterm of other terms, M is either of the form $\mathcal{W}^{-1}(W)$ or the form $\langle N\rangle$.

These observations will be used in the completeness proof. In the following, we simply write $(\_)^{-1}$ for the six inverse translations to avoid clutter.

We can prove that the inverse translation is in fact the (left) inverse of the CPS-translation up to the axioms in Figure 1.

THEOREM 3. *Let M be a term and V be a value in $\lambda_{\mathcal{DA}}$. Then we have:*

$$\begin{array}{rcl} \lambda_{\mathcal{DA}} \vdash ([M])^{-1} & = & M \\ \lambda_{\mathcal{DA}} \vdash (V^*)^{-1} & = & V \end{array}$$

PROOF. We prove the theorem by simultaneous induction on M and V.

(Case M is a value V_1)

$$\begin{array}{rcll} ([V_1])^{-1} & \equiv & \mathcal{D}(\lambda k.\mathcal{A}(k(V_1^*)^{-1})) & \\ & = & \mathcal{D}(\lambda k.\mathcal{A}(kV_1)) & \text{(by I.H.)} \\ & = & \mathcal{D}(\lambda k.kV_1) & \text{(by } \mathcal{D}\text{-abort)} \\ & = & V_1 & \text{(by } \mathcal{D}\text{-current, } \mathcal{D}\text{-elim)} \end{array}$$

(Case $M \equiv N_1N_2$)

$$\begin{array}{rcll} & ([N_1N_2])^{-1} & & \\ \equiv & \mathcal{D}(\lambda k.\mathcal{A}((\lambda m.\langle(\lambda n.\langle k(mn)\rangle)([N_2])^{-1}\rangle)([N_1])^{-1})) & & \\ = & \mathcal{D}(\lambda k.\mathcal{A}((\lambda m.\langle(\lambda n.\langle k(mn)\rangle)N_2\rangle)N_1)) & \text{(by I.H.)} \\ = & \mathcal{D}(\lambda k.\mathcal{A}(\langle k(N_1N_2)\rangle)) & \text{(by } \beta_\Omega\text{-reset-1 twice)} \\ = & \mathcal{D}(\lambda k.\mathcal{A}(k(N_1N_2))) & \text{(by } \mathcal{A}\text{-reset)} \\ = & \mathcal{D}(\lambda k.k(N_1N_2)) & \text{(by } \mathcal{D}\text{-abort)} \\ = & N_1N_2 & \text{(by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim)} \end{array}$$

(Case $M \equiv \langle N\rangle$)

$$\begin{array}{rcll} & ([\langle N\rangle])^{-1} & & \\ \equiv & \mathcal{D}(\lambda k.\mathcal{A}(\mathcal{D}\lambda\gamma.(\lambda v.\gamma\langle kv\rangle)\langle([N]J)^{-1}\rangle)) & & \\ = & \mathcal{D}(\lambda k.\mathcal{A}(\mathcal{D}\lambda\gamma.\gamma\langle k\langle([N]J)^{-1}\rangle\rangle)) & \text{(by } \beta_\Omega\text{-reset-2)} \\ = & \mathcal{D}(\lambda k.\mathcal{A}\langle k\langle([N]J)^{-1}\rangle\rangle) & \text{(by } \mathcal{D}\text{-current, } \mathcal{D}\text{-elim)} \\ = & \mathcal{D}(\lambda k.k\langle([N]J)^{-1}\rangle) & \text{(by } \mathcal{A}\text{-reset,}\mathcal{D}\text{-abort)} \\ = & \langle([N]J)^{-1}\rangle & \text{(by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim)} \\ \equiv & \langle(\lambda x.\langle\mathcal{D}(\lambda\gamma.\gamma x)\rangle)[N]^{-1}\rangle & \\ = & \langle(\lambda x.x)[N]^{-1}\rangle & \text{(by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim,reset-value)} \\ = & \langle[N]^{-1}\rangle & \text{(by } \beta_\Omega\text{)} \\ = & \langle N\rangle & \text{(by I.H.)} \end{array}$$

(Case $V \equiv x$ or $V \equiv \lambda x.M_1$)
Easily shown.
(Case $V \equiv \mathcal{D}$)

$$\begin{array}{rcll} & (\mathcal{D}^*)^{-1} & & \\ \equiv & \lambda x.\mathcal{D}(\lambda k.\mathcal{A}(k(x(\lambda v.\mathcal{D}(\lambda k'.\mathcal{A}(kv)))))) & & \\ = & \lambda x.\mathcal{D}(\lambda k.\mathcal{A}(k(x(\lambda v.kv)))) & \text{(by } \mathcal{D}\text{-abort,}\mathcal{D}\text{-elim)} \\ = & \lambda x.\mathcal{D}(\lambda k.k(xk)) & \text{(by } \eta_v\text{,}\mathcal{D}\text{-abort)} \\ = & \lambda x.\mathcal{D}(\lambda k.xk) & \text{(by } \mathcal{D}\text{-current)} \\ = & \mathcal{D} & \text{(by } \eta_v \text{ twice)} \end{array}$$

(Case $V \equiv \mathcal{A}$)

$$\begin{array}{rcll} (\mathcal{A}^*)^{-1} & \equiv & \lambda x.\mathcal{D}(\lambda k.\mathcal{A}(\mathcal{D}(\lambda\gamma.\gamma x))) & \\ & = & \lambda x.\mathcal{A}x & \text{(by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim twice)} \\ & = & \mathcal{A} & \text{(by } \eta_v\text{)} \end{array}$$

□

5. REDUCTIONS IN THE TARGET CALCULUS

The CPS- and the inverse-translations give a syntactic correspondence between the source and the target calculi. In this section, we establish the correspondence in the equality-level. Namely, we show the equality in the target calculus (the full $\beta\eta$-equality) is preserved through the inverse translation.

5.1 Substitution and Inverse Translation

In this subsection, we prove three lemmas which essentially say that a substitution and the inverse-translation commute. We have three such lemmas corresponding to three classes of variables (an ordinary variable, a continuation variable, and a metacontinuation variable).

The first substitution lemma is for an ordinary variable x.

THEOREM 4. *Let X be one of T-, Q-, R-, W-, K-, G-terms, and W be a W-term.*
Then we have:

$$\lambda_{\mathcal{DA}} \vdash (X\{x := W\})^{-1} = X^{-1}\{x := W^{-1}\}$$

The proof is straightforward.

The second substitution lemma is for a continuation variable k.

THEOREM 5. *Let X be one of T-, R-, W-, G-terms, and Y be one of Q-, K-terms.*

Then we have:

$$\begin{aligned}
\lambda_{\mathcal{DA}} &\vdash (X\{k := K\})^{-1} = X^{-1}\{k := \lambda f.\mathcal{A}(K^{-1}[f])\}\\
\lambda_{\mathcal{DA}} &\vdash \langle (Y\{k := K\})^{-1}\rangle = \langle Y^{-1}\{k := \lambda f.\mathcal{A}(K^{-1}[f])\}\rangle\\
\lambda_{\mathcal{DA}} &\vdash \mathcal{A}(Y\{k := K\})^{-1} = \mathcal{A}(Y^{-1}\{k := \lambda f.\mathcal{A}(K^{-1}[f])\})
\end{aligned}$$

PROOF. The theorem is proved by simultaneous induction on X and Y. We prove only the interesting cases here.

The second equation for $Y \equiv k$ can be proved as follows:

$$\begin{aligned}
LHS &\equiv \langle K^{-1}\rangle\\
RHS &\equiv \langle (\lambda f.\mathcal{A}(K^{-1}[f]))[\ ]\rangle\\
&= \langle \mathcal{A}(K^{-1})\rangle \qquad (\text{by } \beta_\Omega)\\
&= \langle K^{-1}\rangle \qquad (\text{by reset-}\mathcal{A})
\end{aligned}$$

Note that, $\mathcal{A}(K^{-1})$ is a p-context, hence we can apply β_Ω in the above derivation.

The third equation for $Y \equiv k$ can be proved similarly, using $\mathcal{A}$-lift instead of reset-$\mathcal{A}$. □

The third substitution lemma is for a metacontinuation variable γ. Since γ does not appear free in T-, Q-, W-, and K-terms, we need to consider R- and G-terms only.

THEOREM 6. *Let R be an R-term, and G, G_2 be G-terms. Also let M be a term in the source calculus $\lambda_{\mathcal{DA}}$. Then we have:*

$$\begin{aligned}
\lambda_{\mathcal{DA}} &\vdash (R\{\gamma := G_2\})^{-1} = G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.R^{-1})\rangle]\\
\lambda_{\mathcal{DA}} &\vdash (G\{\gamma := G_2\})^{-1}[\langle M\rangle] = G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[\langle M\rangle])\rangle]
\end{aligned}$$

PROOF. The theorem is proved by simultaneous induction on R and G.

(Case: $R \equiv QG$)

Note that Q does not contain γ free.

$$\begin{aligned}
LHS &\equiv (Q(G\{\gamma := G_2\}))^{-1}\\
&\equiv (G\{\gamma := G_2\})^{-1}[\langle Q^{-1}\rangle]\\
&= G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[\langle Q^{-1}\rangle])\rangle] \qquad (\text{by I.H.})\\
RHS &\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.(QG)^{-1})\rangle]\\
&\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[\langle Q^{-1}\rangle])\rangle]
\end{aligned}$$

(Case: $R \equiv GW$)

Note that W does not contain γ free.

$$\begin{aligned}
LHS &\equiv ((G\{\gamma := G_2\})W)^{-1}\\
&\equiv (G\{\gamma := G_2\})^{-1}[W^{-1}]\\
&= (G\{\gamma := G_2\})^{-1}[\langle W^{-1}\rangle] \qquad (\text{by reset-value})\\
&= G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[\langle W^{-1}\rangle])\rangle] \qquad (\text{by I.H.})\\
&= G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[W^{-1}])\rangle] \qquad (\text{by reset-value})\\
RHS &\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.(GW)^{-1})\rangle]\\
&\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.G^{-1}[W^{-1}])\rangle]
\end{aligned}$$

(Case: $G \equiv \gamma$)

$$\begin{aligned}
LHS &\equiv G_2^{-1}[\langle M\rangle]\\
RHS &\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.\gamma\langle M\rangle)\rangle]\\
&= G_2^{-1}[\langle\langle M\rangle\rangle] \qquad (\text{by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim})\\
&= G_2^{-1}[\langle M\rangle] \qquad (\text{by reset-reset})
\end{aligned}$$

(Case: $G \equiv \lambda x.R$)

We may assume $x \notin FV(G_2)$.

$$\begin{aligned}
LHS &\equiv (\lambda x.(R\{\gamma := G_2\})^{-1})\langle M\rangle\\
&= (\lambda x.G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.R^{-1})\rangle])\langle M\rangle \qquad (\text{by I.H.})\\
&= (\lambda x.G_2^{-1}[\langle (\lambda x.\mathcal{D}(\lambda\gamma.R^{-1}))x\rangle])\langle M\rangle \qquad (\text{by } \beta_v)\\
&= G_2^{-1}[\langle (\lambda x.\mathcal{D}(\lambda\gamma.R^{-1}))\langle M\rangle\rangle] \qquad (\text{by } \beta_\Omega\text{-reset-2})\\
&= G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.(\lambda x.R^{-1})\langle M\rangle)\rangle] \qquad (\text{by } \mathcal{D}\text{-tail})\\
RHS &\equiv G_2^{-1}[\langle \mathcal{D}(\lambda\gamma.(\lambda x.R^{-1})\langle M\rangle)\rangle]
\end{aligned}$$

Hence, we are done. □

5.2 The $\beta\eta$-Reductions and the Inverse Translation

We are ready to prove that the $\beta\eta$-equality is preserved by the inverse translation.

THEOREM 7. *Let X_i be T-, R- or W-terms, Y_i be Q- or K-terms, G_i be G-terms in the target calculus, for $i = 1, 2$. Let M be a term in $\lambda_{\mathcal{DA}}$. Then we have the following:*

$$\begin{aligned}
&\lambda_{\beta\eta} \vdash X_1 = X_2 \mathit{implies} \lambda_{\mathcal{DA}} \vdash X_1^{-1} = X_2^{-1}\\
&\lambda_{\beta\eta} \vdash Y_1 = Y_2 \mathit{\ implies} \lambda_{\mathcal{DA}} \vdash \langle Y_1^{-1}\rangle = \langle Y_2^{-1}\rangle, \mathit{and}\\
&\qquad\qquad \lambda_{\mathcal{DA}} \vdash \mathcal{A}\ Y_1^{-1} = \mathcal{A}\ Y_2^{-1}\\
&\lambda_{\beta\eta} \vdash G_1 = G_2 \mathit{implies} \lambda_{\mathcal{DA}} \vdash G_1^{-1}[\langle M\rangle] = G_2^{-1}[\langle M\rangle]
\end{aligned}$$

PROOF. Note that the second equation for Y_i can be derived from the first one for Y_i. Assuming $\lambda_{\mathcal{DA}} \vdash \langle Y_1^{-1}\rangle = \langle Y_2^{-1}\rangle$, we have $\lambda_{\mathcal{DA}} \vdash \mathcal{A}\ Y_1^{-1} = \mathcal{A}\langle Y_1^{-1}\rangle = \mathcal{A}\langle Y_2^{-1}\rangle = \mathcal{A}\ Y_2^{-1}$, by using $\mathcal{A}$-reset twice.

Note also that it suffices to prove the theorem when the equality in the target calculus (the $\beta\eta$-equality) is obtained by one-step β- or η-reduction. Then the theorem is proved by structural induction on X_1, Y_1, and G_1. Again it suffices to prove the term in consideration is the redex, since the inverse translation commutes with hole-filling of contexts, namely, $(C[M])^{-1} = C^{-1}[M^{-1}]$ if we appropriately define C^{-1} for a context C.

We first consider the cases for β-reduction. If the substituted variable is an ordinary variable (x), or a metacontinuation variable (γ), then we can immediately prove these cases from the first and the third substitution lemmas. For instance, let us consider the case where $Y_1 \equiv (\lambda x.Q_3)W$ and $Y_2 \equiv Q_3\{x := W\}$. Then we can prove the second equation as:

$$\begin{aligned}
\langle Y_1^{-1}\rangle &\equiv \langle (\lambda x.\langle Q_3^{-1}\rangle)W^{-1}\rangle\\
&= \langle\langle Q_3^{-1}\{x := W^{-1}\}\rangle\rangle \qquad (\text{by } \beta_v)\\
&= \langle\langle (Q_3\{x := W\})^{-1}\rangle\rangle \qquad (\text{substitution lemma})\\
&= \langle (Q_3\{x := W\})^{-1}\rangle \qquad (\text{by reset-reset})
\end{aligned}$$

The most difficult case is that we substitute a K-term for a continuation variable k, namely, the case where $Y_1 \equiv (\lambda k.Q_3)K$ and $Y_2 \equiv Q_3\{k := K\}$. We can calculate as:

$$
\begin{array}{rcl}
 & & \langle Y_1^{-1}\rangle \\
&\equiv& \langle K^{-1}[\mathcal{D}(\lambda k.\mathcal{A}Q_3^{-1})]\rangle \\
&=& \langle \mathcal{A}(K^{-1}[\mathcal{D}(\lambda k.\mathcal{A}Q_3^{-1})])\rangle \qquad \text{(by reset-}\mathcal{A}\text{)} \\
&=& \langle \mathcal{D}(\lambda k'.\mathcal{A}(K^{-1}[(\lambda k.\mathcal{A}Q_3^{-1})(\lambda f.k'(\mathcal{A}(K^{-1}[f])))]))\rangle \\
& & \qquad \text{(by } \mathcal{D}\text{-lift)} \\
&=& \langle \mathcal{D}(\lambda k'.\mathcal{A}(K^{-1}[\mathcal{A}(Q_3^{-1}\{k := \lambda f.k'(\mathcal{A}(K^{-1}[f]))\})]))\rangle \\
& & \qquad \text{(by } \beta_v\text{)}
\end{array}
$$

We can use $\mathcal{D}$-lift in the derivation above, since $\mathcal{A}(K^{-1})$ is a p-context.

We also calculate a subterm in the last equation as:

$$
\begin{array}{rcl}
 & & \mathcal{A}(Q_3^{-1}\{k := \lambda f.k'(\mathcal{A}(K^{-1}[f]))\}) \\
&=& \mathcal{A}(Q_3^{-1}\{k := \lambda f.\mathcal{A}(K^{-1}[f])\}) \qquad \text{(by } \mathcal{A}\text{-lift)} \\
&=& \mathcal{A}((Q_3\{k := K\})^{-1}) \qquad \text{(by substitution lemma)}
\end{array}
$$

Using this equality, we can proceed the calculation as:

$$
\begin{array}{rcl}
 & & \langle \mathcal{D}(\lambda k'.\mathcal{A}(K^{-1}[\mathcal{A}((Q_3\{k := K\})^{-1})]))\rangle \\
&=& \langle \mathcal{A}(K^{-1}[\mathcal{A}((Q_3\{k := K\})^{-1})])\rangle \qquad \text{(by } \mathcal{D}\text{-elim)} \\
&=& \langle \mathcal{A}((Q_3\{k := K\})^{-1})\rangle \qquad \text{(by } \mathcal{A}\text{-lift)} \\
&=& \langle (Q_3\{k := K\})^{-1}\rangle \qquad \text{(by reset-}\mathcal{A}\text{)} \\
&\equiv& \langle Y_2^{-1}\rangle
\end{array}
$$

In the derivation, $\mathcal{A}$-lift is applicable, since $\mathcal{A}(K^{-1})$ is a p-context.

For the cases of η-reduction, the proofs are almost straightforward. We pick up two cases here.

(Case: $Y_1 \equiv \lambda\gamma.Q_2\gamma$ and $Y_2 \equiv Q_2$)

Let us note that $\gamma \notin FV(Q_2)$.

$$
\begin{array}{rcl}
\langle Y_1^{-1}\rangle &\equiv& \langle \mathcal{D}(\lambda\gamma.\gamma\langle Q_2^{-1}\rangle)\rangle \\
&=& \langle\langle Q_2^{-1}\rangle\rangle \qquad \text{(by } \mathcal{D}\text{-current,}\mathcal{D}\text{-elim)} \\
&=& \langle Q_2^{-1}\rangle \qquad \text{(by reset-reset)}
\end{array}
$$

(Case: $G_1 \equiv \lambda x.G_2 x$ and G_2)

Let M be a term in $\lambda_{\mathcal{DA}}$, then we have:

$$
\begin{array}{rcl}
G_1^{-1}[\langle M\rangle] &\equiv& (\lambda x.G_2^{-1}[x])\langle M\rangle \\
&=& G_2^{-1}[\langle M\rangle] \qquad \text{(by } \beta_\Omega\text{-reset-2)}
\end{array}
$$

□

6. MAIN THEOREMS

In this section, we give the main theorems of this paper.

6.1 Soundness and Completeness of $\lambda_{\mathcal{DA}}$

THEOREM 8 (SOUNDNESS & COMPLETENESS OF $\lambda_{\mathcal{DA}}$). *Let M_1 and M_2 be terms in $\lambda_{\mathcal{DA}}$. Then we have:*

$$\lambda_{\mathcal{DA}} \vdash M_1 = M_2 \text{ if and only if } \lambda_{\beta\eta} \vdash [M_1] = [M_2]$$

PROOF. The soundness (the only-if direction) has already been proved. We prove the completeness (the if-direction). Suppose $\lambda_{\beta\eta} \vdash [M_1] = [M_2]$. Since $[M_i]$ is a T-term (for $i = 1, 2$), we have $\lambda_{\mathcal{DA}} \vdash [M_1]^{-1} = [M_2]^{-1}$ by Theorem 7. Since $\lambda_{\mathcal{DA}} \vdash [M_i]^{-1} = M_i$ (for $i = 1, 2$) by Theorem 3, we have $\lambda_{\mathcal{DA}} \vdash M_1 = M_2$. □

6.2 Soundness and Completeness of $\lambda_{\mathcal{S}}$

So far we have been concentrating on the analysis of the calculus with reset, $\mathcal{D}$- and $\mathcal{A}$-operators. Now we come back to our original motivation, and turn our attention to the calculus with shift and reset. Instead of directly deriving a theory for them as we did for $\lambda_{\mathcal{DA}}$, our strategy here is to relate these two calculi and prove the theory for shift and reset can prove the same set of equations as that for $\lambda_{\mathcal{DA}}$.

To relate $\lambda_{\mathcal{DA}}$ and $\lambda_{\mathcal{S}}$, we define a syntactic translation $(\_)^\circ$ from $\lambda_{\mathcal{DA}}$ terms to $\lambda_{\mathcal{S}}$ terms, which replaces the constants $\mathcal{D}$ and $\mathcal{A}$ by their "definitions" with $\mathcal{S}$. It merely replaces these constants, and preserves other constructs. For the opposite direction, the translation $(\_)^\dagger$ replaces $\mathcal{S}$ by its "definition" with $\mathcal{D}$ and $\mathcal{A}$. We list the interesting cases below:

$$
\begin{array}{rcl}
(\_)^\circ &:& \lambda_{\mathcal{DA}} \to \lambda_{\mathcal{S}} \\
\mathcal{D}^\circ &=& \lambda z.\mathcal{S}(\lambda k.k(z(\lambda x.\mathcal{A}^\circ(kx)))) \\
\mathcal{A}^\circ &=& \lambda z.\mathcal{S}(\lambda d.z)
\end{array}
$$

$$
\begin{array}{rcl}
(\_)^\dagger &:& \lambda_{\mathcal{S}} \to \lambda_{\mathcal{DA}} \\
\mathcal{S}^\dagger &=& \lambda z.\mathcal{D}(\lambda k.\mathcal{A}(z(\lambda x.\langle kx\rangle)))
\end{array}
$$

The following lemma states that these translations are sound with respect to the CPS-translation, and that they are inverses to each other modulo the equality theories. Recall that the CPS-translation for $\lambda_{\mathcal{S}}$ was defined at the same time as that for $\lambda_{\mathcal{DA}}$ in Section 2.

LEMMA 3. *Let M be a term in $\lambda_{\mathcal{DA}}$, and N be a term in $\lambda_{\mathcal{S}}$. Then we have:*

$$
\begin{array}{rcl}
\lambda_{\beta\eta} &\vdash& [M] = [M^\circ] \\
\lambda_{\beta\eta} &\vdash& [N^\dagger] = [N] \\
\lambda_{\mathcal{DA}} &\vdash& M^{\circ\dagger} = M \\
\lambda_{\mathcal{S}} &\vdash& N^{\dagger\circ} = N
\end{array}
$$

PROOF. We only prove the key case of the last equation where $N = \mathcal{S}$.

$$
\begin{array}{rcl}
\mathcal{S}^{\dagger\circ} &\equiv& \lambda z.\mathcal{D}^\circ(\lambda k.\mathcal{A}^\circ(z(\lambda x.\langle kx\rangle))) \\
&=& \lambda z.\mathcal{S}(\lambda k.k(\mathcal{A}^\circ(z(\lambda x.\langle\mathcal{A}^\circ(kx)\rangle)))) \qquad \text{(by } \beta_v\text{)} \\
&=& \lambda z.\mathcal{S}(\lambda k.\mathcal{A}^\circ(z(\lambda x.\langle kx\rangle))) \qquad \text{(by } \mathcal{A}^\circ\text{-lift,reset-}\mathcal{A}^\circ\text{)} \\
&=& \lambda z.\mathcal{S}(\lambda k.\langle z(\lambda x.\langle kx\rangle)\rangle) \qquad \text{(by } \mathcal{S}\text{-reset,reset-}\mathcal{A}^\circ\text{)} \\
&=& \lambda z.\mathcal{S}(\lambda k.z(\lambda x.\langle kx\rangle)) \qquad \text{(by } \mathcal{S}\text{-reset)} \\
&=& \lambda z.\mathcal{S}z \qquad \text{(by } \mathcal{S}\text{-nat)} \\
&=& \mathcal{S} \qquad \text{(by } \eta_v\text{)}
\end{array}
$$

In the derivation above, we used $\mathcal{A}^\circ$-lift ($F[\mathcal{A}^\circ M] = \mathcal{A}^\circ M$) and reset-$\mathcal{A}^\circ$ ($\langle\mathcal{A}^\circ M\rangle = \langle M\rangle$), which follow from reset-$\mathcal{S}$, $\mathcal{S}$-tail, $\mathcal{S}$-nat, and β_v. □

The next lemma states the soundness of these translations.

LEMMA 4. *(i) Let M_1 and M_2 be terms in $\lambda_{\mathcal{DA}}$. Then $\lambda_{\mathcal{DA}} \vdash M_1 = M_2$ implies $\lambda_{\mathcal{S}} \vdash {M_1}^\circ = {M_2}^\circ$.*

(ii) Let N_1 and N_2 be terms in $\lambda_{\mathcal{S}}$. Then $\lambda_{\mathcal{S}} \vdash N_1 = N_2$ implies $\lambda_{\mathcal{DA}} \vdash {N_1}^\dagger = {N_2}^\dagger$.

The proof of this lemma can be found in the appendix.

By combining these two lemmas with Theorem 8, we obtain the completeness of $\lambda_{\mathcal{S}}$.

THEOREM 9 (SOUNDNESS & COMPLETENESS OF $\lambda_{\mathcal{S}}$). *Let N_1 and N_2 be terms in $\lambda_{\mathcal{S}}$. Then we have:*

$$\lambda_{\mathcal{S}} \vdash N_1 = N_2 \text{ if and only if } \lambda_{\beta\eta} \vdash [N_1] = [N_2]$$

PROOF. The soundness has already been proved.

For the completeness, let us suppose $\lambda_{\beta\eta} \vdash [N_1] = [N_2]$. By Lemma 3, $\lambda_{\beta\eta} \vdash [{N_1}^\dagger] = [{N_2}^\dagger]$. By the completeness of $\lambda_{\mathcal{DA}}$, we have $\lambda_{\mathcal{DA}} \vdash {N_1}^\dagger = {N_2}^\dagger$. Then by Lemma 4, $\lambda_{\mathcal{S}} \vdash {N_1}^{\dagger\circ} = {N_2}^{\dagger\circ}$. From this, we can deduce $\lambda_{\mathcal{S}} \vdash N_1 = N_2$ using Lemma 3. □

6.3 Languages with Basic Constants

We remark that our results can be modified to the source calculi which have non-functional (basic) constants, such as natural numbers. One may wonder if our results can be applied to more realistic programming languages, such as Scheme [1], since the axiom η_v does not hold in general in this case. We can, however, give a sound and complete axiomatization for this calculus in the same manner as Sabry and Felleisen's work [25], that is, by simultaneously restricting η_v in the source calculus and η in the target calculus. We omit the details here due to space limitation.

7. IMPLICATIONS OF AXIOMATIZATION AND INDEPENDENCE

In this section, we examine the axioms in $\lambda_{\mathcal{DA}}$ to get some insight on the calculus.

7.1 Conservativity

As we already mentioned in Section 3, the first three axioms β_v, η_v, and β_Ω in Figure 1 correspond to Moggi's computational lambda calculus, and the axioms $\mathcal{D}$-elim, $\mathcal{D}$-current, $\mathcal{D}$-abort, and $\mathcal{D}$-lift correspond to Sabry-Felleisen's axiomatization for callcc if we identify the $\mathcal{D}$-operator with callcc. In other words, the $\mathcal{D}$-operator in $\lambda_{\mathcal{DA}}$ behaves in the same way as callcc[3]. The remaining axioms are needed to axiomatize the delimiter (the reset operator) and the $\mathcal{A}$-operator. All but the axiom reset-lift appear in Sabry's axiomatization for shift and lazy reset [24], but the actual meaning of axioms are not quite the same, since a term in the form $\langle M \rangle$ is a value in his calculus, but not a value in our calculus. Sabry's theory may be obtained by adding axioms to $\lambda_{\mathcal{DA}}$ in an obvious way[4].

An immediate but non-trivial consequence of our axiomatization is that the theory of delimited continuations is a conservative extension of that of first-class (unlimited) continuations, which can be shown below. Since Sabry and Felleisen's axioms [25] for first-class continuations (that is, without delimited continuations) are derivable from our $\lambda_{\mathcal{DA}}$ (where we replace callcc by $\mathcal{D}$), it follows that any valid equation in Sabry and Felleisen's theory is also derivable in $\lambda_{\mathcal{DA}}$. As a reverse direction, let $M = N$ be a provable equation in $\lambda_{\mathcal{DA}}$, where M and N do not contain the reset-operator. Then $M = N$ is provable in Sabry's theory for lazy reset [24], since it is stronger than $\lambda_{\mathcal{DA}}$. By carefully examining Sabry's work, we can see Sabry's theory for shift and lazy reset is conservative over Sabry and Felleisen's theory for first-class continuations (see the discussion in the conclusion below), hence $M = N$ is also provable in Sabry and Felleisen's theory. It thus follows that

THEOREM 10 (CONSERVATIVE EXTENSION). *The theory $\lambda_{\mathcal{DA}}$ is a conservative extension of Sabry and Felleisen's theory of first-class continuations [25] (when we identify $\mathcal{D}$ with* callcc*).*

Note that this is not a trivial observation, as the CPS-translations of the $\mathcal{A}$-operator are not quite the same in these theories (while callcc and $\mathcal{D}$ do have the same translation). However, as one intuitively expects, this difference can be ignored as long as we do not use the reset operator.

7.2 Independence of Axioms

We already mentioned that many theories in the literature (such as Moggi's computational lambda calculus and Sabry and Felleisen's theory for callcc) contain redundant axioms, namely, some axioms are derivable from others. A natural question is whether $\lambda_{\mathcal{DA}}$ consists of independent axioms. We cannot fully answer this question, but as a partial answer, we can show all but one new axioms are independent from other axioms, thus they are not redundant.

In our equational setting, we say an axiom A is independent from a theory $\mathcal{T}$ if A is not derivable from $\mathcal{T}$, and $\mathcal{T} \cup \{A\}$ is consistent, where an equational theory $\mathcal{T}$ is consistent if $M = N$ is not provable from $\mathcal{T}$ for some M and N.

THEOREM 11 (INDEPENDENCE OF AXIOMS). *Let ax be one of $\mathcal{A}$-lift, $\mathcal{A}$-top, $\mathcal{A}$-reset, and reset-value. Then ax is independent from $\lambda_{\mathcal{DA}} - \{ax\}$.*

PROOF. From the soundness theorem (Theorem 1) and the consistency of $\lambda_{\beta\eta}$, we have $\lambda_{\mathcal{DA}}$ is consistent.

To show the underivability of the axioms, we define a modified CPS-translation from $\lambda_{\mathcal{DA}}$ to $\lambda_{\beta\eta}$, under which only one axiom becomes unsound while the others remain sound. For a term M and a value V, we write $(\!|M|\!)$ and $V^\ddagger$ for the modified versions of CPS-translations $[M]$ and V^*, respectively. We will define four modified CPS-translations, but use the same symbols for all the cases.

For the axiom $\mathcal{A}$-lift, we modify the following two definitions of the CPS-translation:

$$\begin{aligned} (\!|\langle M \rangle|\!) &= (\!|M|\!) \\ \mathcal{A}^\ddagger &= \lambda xk.kx \end{aligned}$$

The definitions for other terms are unchanged, for instance, $(\!|V|\!) = \lambda k.\ kV^\ddagger$ and $x^\ddagger = x$. It is easy to see only $\mathcal{A}$-lift is unsound and all the other axioms are sound with respect to this modified CPS-translation, which implies that $\mathcal{A}$-lift is not derivable from other axioms.

For the axiom $\mathcal{A}$-top, we change the following one:

$$(\!|\langle M \rangle|\!) = (\!|M|\!)$$

For the axiom $\mathcal{A}$-reset, we change the following one:

$$(\!|\langle M \rangle|\!) = \lambda k.\ k(\lambda xk'\gamma.\ (\!|M|\!)J(\lambda v.vxk'\gamma))$$

[3]Filinski [11] already mentioned this correspondence.

[4]For each axiom in our theory which refers to a value V, we add a new axiom which is obtained by replacing V by $\langle M \rangle$ in the original axiom.

The term in the right hand side may look rather complex, but it is the result of first expanding the term $[\lambda x.\langle M\rangle x]$, then replacing $[M]$ by $(\!|M|\!)$ (after some β-reductions).

For the axiom reset-value, we change the following one:

$$(\!|\langle M\rangle|\!) = (\!|\mathcal{A}M|\!)$$

In all cases, the unsoundness of the specified axiom and the soundness of all the other axioms can be shown easily. □

We do not know if the remaining axiom reset-lift is independent or not, although we believe so.

8. CONCLUSION

The shift and reset operators attract many researchers' interest because of its expressiveness and rigorous semantics via CPS-translation. However, there are theoretically not pleasant features in these operators. For instance, if we try to have a type system as an extension of the standard simply typed lambda calculus, and we allow the type of the reset terms to be higher-order, then the CPS-translation does not preserve typing, and the calculus is not strongly normalizing. There are several attempts to remedy this defect including Murthy's work [21] and ours [19]. However, due to its simple semantics, most researchers use the original shift and reset operators. In this paper, we return to the original shift and reset, and have given a sound and complete axiomatization for the calculus with shift and reset, and an equivalent calculus with $\mathcal{D}$, $\mathcal{A}$, and reset. The latter calculus $\lambda_{\mathcal{DA}}$ is not only useful for expressing the inverse translation used in the completeness proof, but also interesting on its own right, as it is a conservative extension over Sabry and Felleisen's theory for first-class continuations.

Having a very similar motivation in mind, Sabry [24] axiomatized a calculus with shift and *lazy* reset, which is a close, but different calculus than Danvy and Filinski's. He invented an elegant technique to prove the completeness of such an axiomatization and obtained a complete axiomatization which is very similar to ours. In fact, the only differences of his axioms and ours are that (i) $\langle M\rangle$ is a value in his calculus, and (ii) our calculus has an extra axiom reset-lift. If we regard $\langle M\rangle$ as values, then the axiom reset-lift is derivable, so the essential difference is (i). In practice, we are interested in evaluating the inside of reset-terms, namely, M in the term $\langle M\rangle$ while Sabry's lazy reset suspends evaluation of reset-terms, so the practical importance of his calculus remains uncertain. We have axiomatized the very calculus Danvy and Filinski proposed and many followers have been using, in which sense we can say our axiomatization should be more useful. However, our proof technique is based on Sabry and Felleisen's work [25], and it consists of a lot of syntactic calculations. On the other hand, Sabry's new technique is much more elegant. In order to apply Sabry's proof technique to our case, we have to carefully formalize the target calculus in such a way that the evaluation is in call-by-value in the interpretation of the shift operator, and in call-by-name in manipulating continuation parameters. We did not take this approach in this paper, since it is not clear that the target calculus Danvy and Filinski's originally defined coincides with such a "hybrid" system.

We briefly state some future work. Firstly, we should extend our work to include the full CPS-hierarchy, which involve not only continuation and metacontinuation parameters, but also metameta-continuation parameters and so on (in general, meta^n-continuation parameters for $n > 1$). In this work, we have treated the simplest case of $n = 1$, which means we allow only one kind of shift and reset operators. In some applications [30], one wants to use many different kinds of shift/reset operators which do not interact, thus extending our result to such a general case seems necessary. Since we have used the linearity of the metacontinuation variable γ, a straightforward extension of our proof is not possible for the case of $n > 1$.

Secondly, we have considered equational theories in this paper, and not considered reduction theories, namely, we did not orient equations. As Sabry and Wadler [26] and Barthe et al [3] pointed out, we should also study reduction theories, in particular, we want to know if there is a set of *reduction* rules for shift and reset which is sound and complete with respect to CPS-translation. It is already a difficult problem for the case of first-class continuations. We should at least consider optimized (administrative redex-free) CPS-translation in the sense of Plotkin [22], Steele [28], Danvy-Filinski[6], Sabry-Felleisen [25], and Danvy-Nielsen [8].

Finally, we wish to obtain a deeper, more semantics-oriented understanding of the delimited continuations and our completeness proof. The second author has started to study this direction [16] by replacing the target calculus by (models of) a linear lambda calculus with a recursive type. Let us fix the answer type R, and let D be the recursive type defined by $D = D \to (D \to R_D) \to R_D$ where $R_X = (X \to R) \multimap R$ (the linearly-used continuations monad [15]), then the targets of the CPS-translation in Section 4 can by typed as:

(terms)	$(D \to R_D) \to (D \to R) \multimap R$
(pre-values)	R_D
(answers)	R
(values)	D $(= D \to (D \to R_D) \to R_D)$
(continuations)	$D \to R_D$
(metacontinuations)	$D \to R$

Note that the linearity condition of the metacontinuation variable is expressed by the linear implication $\multimap$. This approach also seems to be closely related to a recent work by Thielecke [31].

Acknowledgments: The authors would like to thank Olivier Danvy and Amr Sabry for careful reading and valuable comments. Thanks are also due to anonymous referees for their insightful suggestions.

9. REFERENCES

[1] H. Abelson et al. Revised^5 Report on the Algorithmic Language Scheme. *Higher-Order and Symbolic Computation*, 11(1):7–105, 1998.

[2] K. Asai. Online Partial Evaluation for Shift and Reset. In *Proc. ACM Workshop on Partial Evaluation and Semantics-Based Program Manipulation*, pages 19–30, 2002.

[3] G. Barthe, J. Hatcliff, and M. H. Sørensen. Reflections on Reflections. In *Proc. Ninth International Symposium on Programming Language Implementation and Logic Programming*, pages 241–258, 1997.

[4] J. Berdine, P. O'Hearn, U. Reddy, and H. Thielecke. Linear Continuation Passing. *Higher-Order and Symbolic Computation*, pages 181–208, 2002.

[5] O. Danvy. Formalizing Implementation Strategies for First-Class Continuations. In *Proc. 9th European Symposium on Computing*, Lecture Notes in Computer Science 1782, pages 151–160, 1990.

[6] O. Danvy and A. Filinski. Abstracting Control. In *Proc. 1990 ACM Conference on Lisp and Functional Programming*, pages 151–160, 1990.

[7] O. Danvy and A. Filinski. Representing Control: a Study of the CPS Transformation. *Mathematical Structures in Computer Science*, 2(4):361–391, 1992.

[8] O. Danvy and L. R. Nielsen. A First-Order One-Pass CPS Transformation. In *Proc. FOSSACS 2002*, Lecture Notes in Computer Science 2303, pages 98–113, 2002.

[9] M. Felleisen. The Theory and Practice of First-Class Prompts. In *Proc. 15th Symposium on Principles of Programming Languages*, pages 180–190, 1988.

[10] M. Felleisen and R. Hieb. The Revised Report on the Syntactic Theories of Sequential Control and State. *Theoretical Computer Science*, 103:235–271, 1992.

[11] A. Filinski. Representing Monads. In *Proc. 21st Symposium on Principles of Programming Languages*, pages 446–457, 1994.

[12] C. Führmann and H. Thielecke. On the Call-by-Value CPS Transform and its Semantics. *Information and Computation*, to appear.

[13] M. Gasbichler and M. Sperber. Final shift for call/cc: Direct Implementation of Shift and Reset. In *Proc. 7th International Conference on Functional Programming*, pages 271–282, 2002.

[14] C. A. Gunter, D. Remy, and J. G. Riecke. A Generalization of Exceptions and Control in ML-Like Languages. In *Proc. Functional Programming and Computer Architecture*, pages 12–23, 1995.

[15] M. Hasegawa. Linearly Used Effects: Monadic and CPS Transformations into the Linear Lambda Calculus. In *Proc. International Symposium on Functional and Logic Programming*, Lecture Notes in Computer Science 2441, pages 167–182, 2002.

[16] M. Hasegawa. On the Semantics of Delimited Continuations. manuscript, 2003.

[17] R. Hieb, R. Dybvig, and C. W. Anderson. Subcontinuations. *Lisp and Symbolic Computation*, 6:453–484, 1993.

[18] M. Hofmann. Sound and Complete Axiomatisations of Call-by-Value Control Operators. *Mathematical Structures in Computer Science*, 5:461–482, 1995.

[19] Y. Kameyama. A Type-Theoretic Study on Partial Continuations. In *Proc. IFIP International Conference on Theoretical Computer Science*, Lecture Notes in Computer Science 1872, pages 489–504, 2000.

[20] E. Moggi. Computational Lambda-Calculus and Monads. In *Proc. 4th Symposium on Logic in Computer Science*, pages 14–28, 1989.

[21] C. Murthy. Control Operators, Hierarchies, and Pseudo-Classical Type Systems: A-Translation at Work. In *Proc. ACM Workshop on Continuation*, pages 49–71, 1992.

[22] G. D. Plotkin. Call-by-Name, Call-by-Value, and the λ-Calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[23] C. Queinnec and B. Serpette. A Dynamic Extent Control Operator for Partial Continuations. In *Proc. 18th Symposium on Principles of Programming Languages*, pages 174–184, 1991.

[24] A. Sabry. Note on Axiomatizing the Semantics of Control Operators. Technical Report CIS-TR-96-03, Dept. of Information Science, Univ. of Oregon, 1996.

[25] A. Sabry and M. Felleisen. Reasoning about Programs in Continuation-Passing Style. *Lisp and Symbolic Computation*, 6(3-4):289–360, 1993.

[26] A. Sabry and P. Wadler. A Reflection on Call-by-Value. *ACM Transactions on Programming Languages and Systems*, 19(6):916–941, 1997.

[27] T. Sekiguchi, T. Sakamoto, and A. Yonezawa. Portable Implementation of Continuation Operators in Imperative Languages by Exception Handling. In *Advances in Exception Handling Techniques*, Lecture Notes in Computer Science 2022, pages 217–233, 2001.

[28] G. L. Steele Jr. RABBIT: A Compiler for SCHEME. Technical Report AITR-474, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1978.

[29] C. Strachey and C. P. Wadsworth. Continuations: A Mathematical Semantics for Handling Full Jumps. *Higher-Order and Symbolic Computation*, 13(1/2):135–152, 2000.

[30] E. Sumii. An Implementation of Transparent Migration on Standard Scheme. In *Proc. Workshop on Scheme and Functional Programming*, pages 61–63, 2000.

[31] H. Thielecke. From Control Effects to Typed Continuation Passing. In *Proc. 30th Symposium on Principles of Programming Languages*, pages 139–149, 2003.

[32] P. Thiemann. Cogen in Six Lines. In *Proc. International Conference on Functional Programming*, pages 180–189, 1996.

APPENDIX

(Proof of Lemma 1)

(β-lift, assuming $x \notin FV(F) \cup FV(M)$)

$$\begin{aligned} & (\lambda x.F[N])M \\ = \ & (\lambda x.(\lambda y.F[y])N)M \qquad (\text{by } \beta_\Omega) \\ = \ & (\lambda x.(\lambda y.F[y])((\lambda x.N)x))M \qquad (\text{by } \beta_v) \\ = \ & (\lambda y.F[y])((\lambda x.N)M) \qquad (\text{by } \beta_\Omega) \\ = \ & F[(\lambda x.N)M] \qquad (\text{by } \beta_\Omega) \end{aligned}$$

(reset-reset)

$$\begin{aligned} \langle\langle M\rangle\rangle &= \langle \mathcal{A}\langle M\rangle\rangle \qquad (\text{by } \mathcal{A}\text{-top}) \\ &= \langle \mathcal{A}M\rangle \qquad (\text{by } \mathcal{A}\text{-reset}) \\ &= \langle M\rangle \qquad (\text{by } \mathcal{A}\text{-top}) \end{aligned}$$

($\mathcal{D}$-top)

$$\begin{array}{rcll}
\langle \mathcal{D}M \rangle & = & \langle \mathcal{A}(\mathcal{D}M) \rangle & \text{(by reset-}\mathcal{A}\text{)} \\
& = & \langle \mathcal{D}(\lambda k.\mathcal{A}(M(\lambda f.k(\mathcal{A}f)))) \rangle & \text{(by } \mathcal{D}\text{-lift)} \\
& = & \langle \mathcal{D}(\lambda k.\mathcal{A}(M(\lambda f.\mathcal{A}f))) \rangle & \text{(by } \mathcal{D}\text{-abort)} \\
& = & \langle \mathcal{A}(M(\lambda f.\mathcal{A}f)) \rangle & \text{(by } \mathcal{D}\text{-elim)} \\
& = & \langle M(\lambda f.\mathcal{A}f) \rangle & \text{(by reset-}\mathcal{A}\text{)} \\
& = & \langle M\mathcal{A} \rangle & \text{(by } \eta_v\text{)}
\end{array}$$

($\mathcal{D}$-tail)
Putting $F \equiv [\ ]$ in $\mathcal{D}$-lift, we have $\mathcal{D}M = \mathcal{D}(\lambda k.Mk)$. Then we can derive:

$$\begin{array}{rcll}
& & (\lambda x.\mathcal{D}(\lambda k.M_1))M_2 & \\
& = & \mathcal{D}((\lambda x.\lambda k.M_1)M_2) & \text{(by } \beta\text{-lift)} \\
& = & \mathcal{D}(\lambda z.(\lambda x.\lambda k.M_1)M_2 z) & \text{(by the above equation)} \\
& = & \mathcal{D}(\lambda z.(\lambda k.(\lambda x.M_1)M_2)z) & \text{(by } \beta\text{-lift)} \\
& = & \mathcal{D}(\lambda k.(\lambda x.M_1)M_2) & \text{(by } \beta_v\text{)}
\end{array}$$

(β_Ω-reset-1)

$$\begin{array}{rcll}
& & \langle (\lambda x.\langle F[x] \rangle)M \rangle & \\
& = & \langle (\lambda x.\mathcal{D}(\lambda k.k\langle F[x] \rangle))M \rangle & \text{(by } \mathcal{D}\text{-elim,}\mathcal{D}\text{-current)} \\
& = & \langle \mathcal{D}(\lambda k.(\lambda x.k\langle F[x] \rangle)M) \rangle & \text{(by } \mathcal{D}\text{-tail)} \\
& = & \langle \mathcal{A}(\mathcal{D}(\lambda k.(\lambda x.k\langle F[x] \rangle)M)) \rangle & \text{(by reset-}\mathcal{A}\text{)} \\
& = & \langle \mathcal{D}(\lambda k.(\lambda x.(\lambda f.k(\mathcal{A}f))\langle F[x] \rangle)M) \rangle & \text{(by } \mathcal{D}\text{-lift)} \\
& = & \langle \mathcal{D}(\lambda k.(\lambda x.(\lambda f.\mathcal{A}f)\langle F[x] \rangle)M) \rangle & \text{(by } \mathcal{D}\text{-abort)} \\
& = & \langle (\lambda x.\mathcal{A}(\langle F[x] \rangle))M \rangle & \text{(by } \mathcal{D}\text{-elim,}\eta_v\text{)} \\
& = & \langle (\lambda x.\mathcal{A}(F[x]))M \rangle & \text{(by } \mathcal{A}\text{-reset)} \\
& = & \langle \mathcal{A}(F[M]) \rangle & \text{(by } \beta_\Omega\text{)} \\
& = & \langle F[M] \rangle & \text{(by reset-}\mathcal{A}\text{)}
\end{array}$$

(β_Ω-reset-2)
We prove $(\lambda x.E[x])\langle M \rangle = E[\langle M \rangle]$ by induction on E. The only interesting case is $E = \langle E_1 \rangle$.

$$\begin{array}{rcll}
& & (\lambda x.E[x])\langle M \rangle & \\
& \equiv & (\lambda x.\langle E_1[x] \rangle)\langle M \rangle & \\
& = & \langle (\lambda x.E_1[x])\langle M \rangle \rangle & \text{(by reset-lift)} \\
& = & \langle E_1[\langle M \rangle] \rangle & \text{(by I.H.)} \\
& \equiv & E[\langle M \rangle] &
\end{array}$$

(Proof of Lemma 2)
We can prove $\mathcal{S}$-nat as follows:

$$\begin{array}{rcll}
F[\mathcal{S}M] & = & \mathcal{S}(\lambda k.k(F[\mathcal{S}M])) & \text{(by } \mathcal{S}\text{-elim)} \\
& = & \mathcal{S}(\lambda k.\langle k(F[\mathcal{S}M]) \rangle) & \text{(by } \mathcal{S}\text{-reset)} \\
& = & \mathcal{S}(\lambda k.\langle M(\lambda x.\langle k(F[x]) \rangle) \rangle) & \text{(by reset-}\mathcal{S}\text{)} \\
& = & \mathcal{S}(\lambda k.M(\lambda x.\langle k(F[x]) \rangle)) & \text{(by } \mathcal{S}\text{-reset)}
\end{array}$$

Other equations follow from $\mathcal{S}$-nat.
(Proof of Lemma 4)
We can prove that each axiom in $\lambda_{\mathcal{DA}}$ is translated to a provable equation in $\lambda_{\mathcal{S}}$, and vice versa. Here we give a proof of the most interesting case, namely, we will verify the axiom $\mathcal{D}$-lift is translated to a provable equation in $\lambda_{\mathcal{S}}$.

$$\begin{array}{rcll}
& & (F[\mathcal{D}M])^\circ & \\
& \equiv & F^\circ[\mathcal{D}^\circ M^\circ] & \\
& = & F^\circ[\mathcal{S}(\lambda k.k(M^\circ(\lambda x.\mathcal{A}^\circ(kx))))] & \text{(by } \mathcal{S}\text{-tail,}\beta_\Omega\text{)} \\
& = & \mathcal{S}(\lambda h.(\lambda k.k(M^\circ(\lambda x.\mathcal{A}^\circ(kx))))(\lambda x.\langle hF^\circ[x] \rangle)) & \\
& & \text{(by } \mathcal{S}\text{-nat)} & \\
& = & \mathcal{S}(\lambda h.(\lambda x.\langle hF^\circ[x] \rangle)(M^\circ(\lambda x.\mathcal{A}^\circ\langle hF^\circ[x] \rangle))) & \\
& & \text{(by } \beta_v\text{)} & \\
& = & \mathcal{S}(\lambda h.(\lambda x.hF^\circ[x])(M^\circ(\lambda x.\mathcal{A}^\circ\langle hF^\circ[x] \rangle))) & \\
& & \text{(by } \mathcal{S}\text{-lift-reset)} & \\
& = & \mathcal{S}(\lambda h.hF^\circ[M^\circ(\lambda x.\mathcal{A}^\circ\langle hF^\circ[x] \rangle)]) & \text{(by } \beta_\Omega\text{)} \\
& = & (\mathcal{D}(\lambda h.F[M(\lambda x.hF[x])]))^\circ & \text{(by } \beta_v, \beta_\Omega\text{)}
\end{array}$$

In the derivation we used the following equation as $\mathcal{S}$-lift-reset:

$$\mathcal{S}(\lambda k.(\lambda x.\langle N \rangle)M) = \mathcal{S}(\lambda k.(\lambda x.N)M)$$

which can be proved by $\mathcal{S}$-tail and $\mathcal{S}$-reset.