Axiomatizing Higher Level Delimited Continuation

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

ABSTRACT

In our previous work we gave a sound and complete axiomatization of the control operators for delimited continuations, **shift** and **reset** by Danvy and Filinski and their variants. Since the calculus allows only one use of shift and reset, a next step is to investigate the calculus with many different **shift**'s and **reset**'s. In this work, we study the calculus with higher-level delimited continuation operators, and give a sound and complete axiomatization of the calculus with level-1 and level-2 control operators for delimited continuations.

Due to lack of space, we leave the detailed proof to a separate draft available at:

`http://logic.is.tsukuba.ac.jp/~kam/level2-proof.ps`

1. INTRODUCTION

We are interested in precise semantics of control operators such as **callcc** in Scheme and SML/NJ. Such semantics is often given by a set of equations (axioms) which characterize the operational behavior of control operators. In our previous work [6], we axiomatized the control operators for delimited continuations. Specifically, we gave sound and complete axiomatizations to the following two calculi:

- shift and reset by Danvy and Filinski [1]
- callcc, abort and reset

These two theories are equivalent in the sense that we have a natural interpretation from one theory to the other. For the latter case, the theory consists of (1) the axioms in computational lambda calculus by Moggi [7], (2) the axioms for callcc and abort by Sabry and Felleisen [10], and (3) a few simple and natural axioms which characterizes the reset operator. Moreover, we showed the latter theory is conservative over the theory for callcc and abort, hence, the extra expressiveness of delimited continuations is exactly that expressed by the delimiter (as one expects.)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2002 ACM X-XXXXX-XX-X/XX/XX ...$5.00.

While the shift and reset operators give programmers more freedom to control their programs, a problem arises when one wants to combine two programs each of which has *different use* of shift and reset.

To see this, let us consider the following situation. We are to prove the correctness of partial evaluator, which receives a source program and after some binding-time analysis, which evaluates static parts of the program and produces a residue program. Several partial evaluators use the shift and reset operators to perform so called let-insertion (see for example Thiemann [11]). If the source program of partial evaluator already contains the shift and reset operators and those operators are static in the sense that they must be evaluated at the specialization time, then a conflict may arise. The two different uses of shift and reset operators might interact so that shift in the source program may capture a continuation up to a reset operator for let-insertion, which would result in a wrong answer.

A solution of this problem already exists in Danvy and Filinski's work [1]. Namely, they proposed an infinite number of shift and reset operators indexed by a natural number $n > 0$, and each level-n shift operator captures the continuation up to the (closest) reset operator of level-n or higher. To explain it in the evaluation-context semantics, let E_1 be an arbitrary evaluation-context, and E_2 be an evaluation-context with level less than n, and $m \geq n$. Then we have:

$$E_1[\langle E_2[\xi_n c.M]\rangle_m]$$
evaluates to
$$E_1[\langle M\{c := \lambda f.\langle E_2[f]\rangle_n\}\rangle_m]$$

where $\langle\_\rangle_m$ is the level-m reset operator, $\xi_n c.M$ is the level-n shift operator, and $\{c := \cdots\}$ is the usual capture-avoiding substitution. It is easy to see the shift operator captures the delimited continuation up to the corresponding reset operator. A subtle point here is that level-n shift cannot escape from level-m reset if $m \geq n$, thus each shift cannot freely select its corresponding reset. Rather, higher-level operators dominate lower-level operators, thus the *hierarchy* of control operators.

Surprisingly, this hierarchy of shift and reset has a (relatively) simple and uniform functional CPS-transform which Danvy and Filinski called CPS-hierarchy. The CPS-hierarchy may be viewed as a composition of n CPS-translations, thus inserting n continuation parameters. In order to consider level-n control operators, we need $n+1$ continuation parameters, for instance, the standard shift operator (level-1 shift) needs the CPS-hierarchy for two continuation parameters. For details about CPS-hierarchy, see Danvy and Filinski [1]

and Danvy and Yang [2].

The use of level-n delimited control operators is not restricted to program manipulations. A certain kinds of control mechanism such as backtracking can be written concisely with shift and reset, and to build a large program from small modules, one needs to ensure each use of shift and reset does not interfere. The hierarchy of level-n control operators can give a (partial) solution to such a situation. Note, however, since higher levels dominate lower levels, each use of shift and reset cannot be completely independent, thus we cannot freely compose two program modules both using shift and reset. The hierarchy of level-n shift and reset operators is useful in the case where Danvy and Yang [2] mention other applications of higher level delimited continuations, hierarchical backtracking and quantifier alternation.

One may consider other variations for delimited- (or sometimes called partial-) continuation operators, such as Felleisen's prompt [3] and Hieb et al's subcontinuations [9]. There are also extensions of these operators which allows multiple uses of control operators [4], [5]. These proposals are certainly useful and easier to use than higher-level shift and reset. However, an important merit of Danvy and Filinski's operators is that they have a simple, functional CPS-transform, while other proposals do not have such simple (and functional) CPS-transform as far as we know. To our view, the CPS-transform best describe the precise semantics of control operators, thus, we can build a solid foundation on top of them.

In this work, we first discuss the difficulty of axiomatizing the level-n delimited continuations, then extend our previous result to the level-2 world, namely, we give a sound and complete axiomatization of the calculus with level-1 and level-2 control operators for delimited continuations.

2. DIFFERENCES BETWEEN LEVEL-1 AND LEVEL-2 AXIOMATIZATION

One may think extending the axiomatization of level-1 to level-2 can be done straightforwardly. But there are several differences between level-1 operators and level-n ($n > 1$) operators and these differences made our task harder than one might expect initially. Let us explain them in order.

Firstly, the level-2 abortive operator $\mathcal{A}_2$ cannot be considered as a constant (value). In the case of the level-1 abortive operator $\mathcal{A}_1$, the equation $(\lambda x.\mathcal{A}_1 x)M = \mathcal{A}_1 M$ is valid, so introducing this operator as a term constructor is equivalent to have it as a value corresponding to $\lambda x.\mathcal{A}_1 x$. On the other hand, $(\lambda x.\mathcal{A}_2 x)M = \mathcal{A}_2 M$ is in general not valid with respect to the CPS-semantics (for instance, let M be $\mathcal{A}_1 N$ then the left hand side is equal to $\mathcal{A}_1 N$ (since $\lambda x.\mathcal{A}_2 x$ is a value) while the right hand side is equal to $\mathcal{A}_2 N$ (since $\mathcal{A}_2$ is stronger than $\mathcal{A}_1$). As a conclusion, we should take care of the difference between $\mathcal{A}_2$ and $\lambda x.\mathcal{A}_2 x$. (We also note that, the term $\lambda x.\mathcal{A}_2 x$ appears in several places in our proof.)

Secondly, and more importantly, the callcc-lift axiom does not extend naturally to the case of level-2. To see this, let us recall the axiom for level-1:

$$E_0[\mathsf{callcc} M] = \mathsf{callcc}(\lambda k.E_0[M(\lambda f.k(E_0[f]))])$$

where E_0 is a pure evaluation context (level-0 evaluation context), $k \notin FV(E_0[M])$ and $f \notin FV(kE_0[\ ])$. This axiom (or its variant) has been the central axiom to characterize the role of callcc operator. However, its level-2 counterpart:

$$E_1[\mathsf{callcc}_2 M] = \mathsf{callcc}_2(\lambda k.E_1[M(\lambda f.k(E_1[f]))])$$

is not valid when we take E_1 to be a level-1 evaluation context[1]. For instance, let E_1 be $\mathcal{A}_1[\ ]$ (where $\mathcal{A}_1$ is the level-1 abort operator), then

$$\begin{aligned} LHS &= \mathcal{A}_1(\mathsf{callcc}_2 M) = \mathsf{callcc}_2 M \\ RHS &= \mathsf{callcc}_2(\lambda k.E_1[M(\lambda f.k(\mathcal{A}_1 f))]) \\ &= \mathsf{callcc}_2(\lambda k.E_1[M(\lambda f.\mathcal{A}_1 f)]) \\ &= E_1[M(\lambda f.\mathcal{A}_1 f)] \end{aligned}$$

The results are completely different. Lack of the callcc-lift axiom brings a big difficulty in obtaining a complete axiomatization.

A rescue for this problem is given by Murthy's "telescope" axiom [8]. A level-2 instance of this axiom is:

$$\mathcal{C}_2(\lambda\gamma.M) = \mathcal{C}_1(\lambda x.\mathcal{C}_2(\lambda y.M\{\gamma := \lambda v.y\langle xv\rangle_1\}))$$

where we used the $\mathcal{C}$-operator instead of callcc since the axioms looks simpler for the case of $\mathcal{C}$-operator than callcc[2] The role of this axiom is to split a single occurrence of level-n control operator ($\mathcal{C}_2$ in this case) to a sequence of n control operators of level $1, 2, \cdots, n$.

Combining the telescope axiom with the level-1 $\mathcal{C}$-lift axiom (which is valid), we can simulate the level-2 version of $\mathcal{C}$-lift axiom as:

$$\begin{aligned} & \langle E[\mathcal{C}_2(\lambda\gamma.M)]\rangle_1 \\ = \ & \langle E[\mathcal{C}_1(\lambda k.\mathcal{C}_2(\lambda\gamma.M\{\gamma := \lambda v.\gamma\langle kv\rangle_1\}))]\rangle_1 \\ = \ & \langle \mathcal{C}_1(\lambda k.(\mathcal{C}_2(\lambda\gamma.M\{\gamma := \lambda v.\gamma\langle k(\mathcal{A}_1(E[v]))\rangle_1\}))\rangle_1 \\ = \ & \langle \mathcal{C}_1(\lambda k.\mathcal{C}_2(\lambda\gamma.M\{\gamma := \lambda v.\gamma\langle E[v]\rangle_1\}))\rangle_1 \\ = \ & \langle \mathcal{C}_2(\lambda\gamma.M\{\gamma := \lambda v.\gamma\langle E[v]\rangle_1\})\rangle_1 \\ = \ & \mathcal{C}_2(\lambda\gamma.M\{\gamma := \lambda v.\gamma\langle E[v]\rangle_1\}) \end{aligned}$$

where E is a level-0 evaluation context.

We can easily see that the outer context $\langle E[\ ]\rangle_1$ was captured and passed to the parameter γ. This rewriting has the same effect as $\mathcal{C}$-lift except that we cannot lift $\mathcal{C}_2$ step by step, but we must lift it to the next reset operator at a time.

3. HIGHER LEVEL SHIFT AND RESET

Our source language is a type-free lambda calculus with control operators with the call-by-value semantics. Our axiomatization should go through for the typed case, but it is left for future work. In this paper, we restrict ourselves to level-2 only. The grammar of the source calculus is given by:

$$\begin{aligned} \text{(terms)} \quad M, N &::= V \mid MN \mid \langle M\rangle_1 \mid \langle M\rangle_2 \\ \text{(values)} \quad V &::= x \mid \lambda x.M \mid \mathcal{C}_1 \mid \mathcal{C}_2 \end{aligned}$$

where $\langle\_\rangle_i$ is the reset operator and $\mathcal{C}_i$ is a variant of the shift operator (similar to Felleisen's operator). The original shift operator proposed by Danvy and Filinski can be defined

[1]Here level-i context is an evaluation context in which there are no reset operators of level-$i+1$ or higher which enclose the hole.

[2]This choice is only superficial, since a similar but more complex axiom holds for callcc.

by $\mathcal{C}_i$. The index i indicates the level, with $i = 1$ being the base level. We abbreviate $\mathcal{C}_i(\lambda x.M)$ as $\mathcal{C}_i x.M$, and if $x \notin FV(M)$, it is also abbreviated as $\mathcal{A}_i M$.

The informal semantics of these control operators is given using evaluation contexts as follows (we assume $i > j$ here):

$$\begin{array}{rcl} E[\langle V\rangle_i] & \to & E[V] \\ E[\langle E^{j-1}[\mathcal{C}_j V]\rangle_i] & \to & E[\langle V(\lambda f.\mathcal{A}_j(E^{j-1}[f]))\rangle_i] \\ E[\langle E^{j-1}[\mathcal{A}_j V]\rangle_i] & \to & E[V] \end{array}$$

where E is an evaluation context (in call-by-value), and E^j is a level-j evaluation context defined below.

The first line says delimiting a value does nothing. The second line explains how $\mathcal{C}_j$ works; it captures the continuation up to (delimited by) the reset operator whose level is the same as or higher than j. Unlike the shift operator, but like callcc, the operator $\mathcal{C}_j$ inserts an occurrence of $\mathcal{A}_j$ in the captured continuation. The third line says that $\mathcal{A}_j$ aborts the current continuation up to the reset operator whose level is the same as j or higher.

The evaluation contexts used above are defined as follows:

$$\begin{array}{rcl} E & = & [\,] \mid EM \mid VE \mid \langle E\rangle_j \\ E^i & = & [\,] \mid E^i M \mid V E^i \mid \langle E^i\rangle_j \quad \text{for } j \le i \end{array}$$

We can also consider the shift operator. The operational semantics of the control operator $\mathcal{S}$ is given by:

$$E[\langle E^{j-1}[\mathcal{S}_j V]\rangle_i] \to E[\langle V(\lambda f.\langle E^{i-1}[f]\rangle_j)\rangle_i]$$

Then the real shift operator is given by $\xi_i c.M \equiv \mathcal{S}_i(\lambda c.M)$. The two operators $\mathcal{C}$ and $\mathcal{S}$ can be defined by each other:

$$\begin{array}{rcl} \mathcal{S}_i & = & \lambda z.\mathcal{C}_i(\lambda k.z(\lambda x.\langle kx\rangle_i)) \\ \mathcal{C}_i & = & \lambda z.\mathcal{S}_i(\lambda k.z(\lambda x.\mathcal{S}_i(\lambda d.kx))) \end{array}$$

4. CPS-TRANSLATION

A CPS-translation is a syntactic translation from the source calculus given in the previous section to the target calculus, the pure type-free lambda calculus without control operators.

The strength of shift and reset operators due to Danvy and Filinski is their semantics is precisely given by a functional CPS-translation. Unlike ordinary CPS-translations, their CPS-translation takes more than one continuation parameters in order to interpret level-n control operators. More precisely, if the source calculus has up to level-n control operators, we need a CPS-transform with $n + 1$ continuation parameters.

In our case, we have up to level-2 control operators, so we need CPS-hierarchy with three continuation parameters, which are denoted by k, γ, and δ. These three continuation parameters are called the (standard) continuation parameter, the meta- continuation parameter, and the meta-meta-continuation parameter, respectively.

In summary, the CPS-translation here translates a term in the source calculus (with control operators) to a term in the target calculus, and the resulting term takes three parameters k, γ, δ. For readability we present here a CPS-translation where all the η-redexes are reduced, so some resulting terms may not take three parameters.

Here we give a Plotkin-style CPS-translation, that is, for the translation of a function, the continuation parameter comes after the argument of the function.

$$\begin{array}{rcl} [\![\_]\!] & : & \text{Term} \to \text{Target Term} \\ [\![V]\!] & = & \lambda k.kV^* \\ [\![MN]\!] & = & \lambda k.[\![M]\!](\lambda m.[\![N]\!](\lambda n.mnk)) \\ [\![\langle M\rangle_1]\!] & = & \lambda k\gamma.[\![M]\!]\theta_1(\lambda v.kv\gamma) \\ [\![\langle M\rangle_2]\!] & = & \lambda k\gamma\delta.[\![M]\!]\theta_1\theta_2(\lambda v.kv\gamma\delta) \end{array}$$

$$\begin{array}{rcl} (\_)^* & : & \text{Value} \to \text{Target Term} \\ x^* & = & x \\ (\lambda x.M)^* & = & \lambda x.[\![M]\!] \\ {\mathcal{C}_1}^* & = & \lambda x k.x(\lambda v k'.kv)\theta_1 \\ {\mathcal{C}_2}^* & = & \lambda x k\gamma.x(\lambda v k'\gamma'.kv\gamma)\theta_1\theta_2 \end{array}$$

where we used abbreviations as $\theta_1 = \lambda x\gamma.\gamma x$ and $\theta_2 = \lambda x\delta.\delta x$.

5. AXIOMS

The set of axioms for $\lambda\mathcal{C}^2$ is given in Figure 1 where we introduce an abbreviation $\langle M\rangle_0 \equiv M$ to state the axiom $\mathcal{C}$-elim uniformly.

In order to read the axioms, recall that we abbreviate $\mathcal{C}_i(\lambda x.M)$ as $\mathcal{C}_i x.M$. Note also that the indices i and j range over $\{1, 2\}$ only, hence, we do not consider the case for $j = 0$ in, e.g., the axiom reset-lift-2 (such an instance is not sound).

Most axioms are level-i extension of those for level-1 in our previous work [6]. (1) The axiom β_Ω is valid only for level-0 contexts. To compensate the weakness of this axiom, we have two axioms, reset-lift and reset-lift-2. (2) The axiom $\mathcal{C}$-lift is only applicable to level-0, as we explained before. (3) Most other axioms but telescope are natural extensions from the axioms for level-1. However, there are subtle differences, for instance, the axiom $\mathcal{C}$-elim needs an occurrence of a reset operator if it is applied to level-2. In the $\mathcal{C}$-abort axiom, the argument of k is restricted to a value V, while its level-1 counterpart allows an arbitrary term. The axiom such as reset-reset and $\mathcal{C}$-top were not included in the level-1 axiomatization because their level-1 instances are derivable from other axioms. Since it is not known to us if their level-2 instances are derivable or not, we included these axioms. (4) The axiom reset-$\mathcal{C}$ and telescope did not exist in the level-1 for the obvious reason.

The telescope axiom (written in terms of the shift operator) was first proposed by Murthy [8]. We should also remark that Murthy gave a type system and based on the typing he derived many sound equations, most of which are very similar to ours, but he did not mention completeness. The precise relationship between his axioms and ours is not clear at present, since he gave axioms based on the $\mathcal{S}_i$ operator and also he worked in a typed setting. However, both theories look quite similar; in a sense, our present work could be understood as proving the completeness of (some variation of) Murthy's axioms.

The main result of this paper is that our theory for level-2 is sound and complete with respect to the CPS-translation.

THEOREM 1 (SOUNDNESS & COMPLETENESS). *Let* M_1

$$
\begin{array}{rll}
(\lambda x.M)V = M\{x := V\} & & \beta_v \\
\lambda x.\ Vx = V & & \eta_v, \text{ if } x \notin FV(V) \\
(\lambda x.E^0[x])M = E^0[M] & & \beta_\Omega, \text{ if } x \notin FV(E^0) \\
\langle V\rangle_i = V & & \text{reset-value} \\
\langle(\lambda x.M)\langle N\rangle_i\rangle_i = (\lambda x.\langle M\rangle_i)\langle N\rangle_i & & \text{reset-lift} \\
\langle(\lambda x.M)\langle N\rangle_j\rangle_i = \langle(\lambda x.\langle M\rangle_i)\langle N\rangle_j\rangle_i & & \text{reset-lift-2, } i > j \\
\\
\mathcal{C}_i k.\langle M\rangle_i = \mathcal{C}_i k.M & & \mathcal{C}\text{-reset} \\
\mathcal{C}_i k.k\langle M\rangle_{i-1} = \langle M\rangle_{i-1} & & \mathcal{C}\text{-elim, if } k \notin FV(M) \\
\mathcal{C}_i k.C[E^i[kV]] = \mathcal{C}_i k.C[kV] & & \mathcal{C}\text{-abort, if } k \text{ is not captured by } C \\
E^0[\mathcal{C}_1 M] = \mathcal{C}_1 k.M(\lambda f.k(E^0[f])) & & \mathcal{C}\text{-lift, if } k \notin FV(E^0[M]) \text{ and } f \notin FV(kE^0) \\
\langle\langle M\rangle_i\rangle_j = \langle M\rangle_{max(i,j)} & & \text{reset-reset} \\
\langle\mathcal{C}_i k.M\rangle_j = \mathcal{C}_i k.M & & \text{reset-}\mathcal{C}\text{, if } i > j \\
\langle\mathcal{C}_i k.M\rangle_i = \langle M\{k := \lambda x.\mathcal{C}_i k'.x\}\rangle_i & & \mathcal{C}\text{-top} \\
\mathcal{C}_2\gamma.M = \mathcal{C}_1 k.\mathcal{C}_2\gamma.M\{\gamma := \lambda f.\gamma\langle kf\rangle_1\} & & \text{telescope}
\end{array}
$$

Figure 1: Axioms of $\lambda\mathcal{C}^2$ (i, j range over $\{1, 2\}$)

and M_2 be terms in the source calculus. Then we have:

$$\lambda\mathcal{C}^2 \vdash M_1 = M_2 \text{ if and only if } \lambda_{\beta\eta} \vdash [\![M_1]\!] = [\![M_2]\!]$$

We can prove the soundness by calculation. In the next section we give a very brief summary of completeness proof. The details can be found in the accompanying draft.

6. COMPLETENESS PROOF

As in our previous work, we first analyze the structure of the target calculus then define the translation from the target to the source. The target calculus is the closure of the image of the CPS-translation which is closed by the $\beta\eta$-reductions.

$$
\begin{array}{rcl}
\text{(term) } T & ::= & \lambda k.Q \mid WW \\
\text{(preterm-1) } Q & ::= & \lambda\gamma.R \mid TK \mid KW \\
\text{(preterm-2) } R & ::= & \lambda\delta.S \mid QG \mid GW \\
\text{(answer) } S & ::= & RH \mid HW \\
\text{(value) } W & ::= & x \mid \lambda x.T \\
\text{(continuation) } K & ::= & k \mid \lambda x.Q \\
\text{(meta-cont.) } G & ::= & \gamma \mid \lambda x.R \\
\text{(meta-meta-cont.) } H & ::= & \delta \mid \lambda x.S
\end{array}
$$

Terms in the class T are called T-terms. Similarly for Q-, R-, S-, W-, K-, G-, H-terms. Terms, values, level-0 context (i.e. E^0), level-1 context (i.e. E^1), and level-2 context (i.e. E) are mapped to T-, W-, K-, G-, H-terms in the target calculus. Q- and R-terms do not have intuitive counterpart in the source calculus. We should note that the meta-meta-continuation variable δ is linear.

The inverse translation from the target language to the source language is given by:

$$
\begin{array}{rcl}
\mathcal{T}^{-1}(\lambda k.Q) & = & \mathcal{C}_1(\lambda k.\mathcal{Q}^{-1}(Q)) \\
\mathcal{T}^{-1}(W_1W_2) & = & \mathcal{W}^{-1}(W_1)\mathcal{W}^{-1}(W_2) \\
\mathcal{Q}^{-1}(\lambda\gamma.R) & = & \mathcal{C}_2(\lambda\gamma.\mathcal{R}^{-1}(R)) \\
\mathcal{Q}^{-1}(TK) & = & \mathcal{K}^{-1}(K)\mathcal{T}^{-1}(T) \\
\mathcal{Q}^{-1}(KW) & = & \mathcal{K}^{-1}(K)\mathcal{W}^{-1}(W) \\
\mathcal{R}^{-1}(\lambda\delta.S) & = & \mathcal{S}^{-1}(S) \\
\mathcal{R}^{-1}(QG) & = & \mathcal{G}^{-1}(G)\langle\mathcal{Q}^{-1}(Q)\rangle_1 \\
\mathcal{R}^{-1}(GW) & = & \mathcal{G}^{-1}(G)\mathcal{W}^{-1}(W) \\
\mathcal{S}^{-1}(RH) & = & \mathcal{H}^{-1}(H)\langle\mathcal{R}^{-1}(R)\rangle_2 \\
\mathcal{S}^{-1}(HW) & = & \mathcal{H}^{-1}(H)\mathcal{W}^{-1}(W) \\
\mathcal{W}^{-1}(x) & = & x \\
\mathcal{W}^{-1}(\lambda x.T) & = & \lambda x.\mathcal{T}^{-1}(T) \\
\mathcal{K}^{-1}(k) & = & k \\
\mathcal{K}^{-1}(\lambda x.Q) & = & \lambda x.\langle\mathcal{Q}^{-1}(Q)\rangle_1 \\
\mathcal{G}^{-1}(\gamma) & = & \gamma \\
\mathcal{G}^{-1}(\lambda x.R) & = & \lambda x.\langle\mathcal{R}^{-1}(R)\rangle_2 \\
\mathcal{H}^{-1}(\delta) & = & \lambda x.x \\
\mathcal{H}^{-1}(\lambda x.S) & = & \lambda x.\mathcal{S}^{-1}(S)
\end{array}
$$

Since the meta-meta-continuation variable δ is linear, we can simply ignore it. It can be seen that the above inversion takes a very uniform form, and we can extend it to level-3 or higher levels.

The remaining proof proceeds as in the previous work. (1) Prove the inverse translation is really an inverse of the CPS-translation (up to the equality in the source calculus), that is, $\mathcal{T}^{-1}([\![M]\!]) = M$ in $\lambda\mathcal{C}^2$. (2) Prove the β- and η-equality in the target are preserved by the inverse translation. (3) To prove (2), we need to prove substitutions in the target calculus is preserved by the inverse translation.

The detailed proof needs a lot of calculations so we leave the detailed description to a separate draft.

7. CONCLUSION

We have given the axiomatization of the level-2 delimited continuation operators, thus we can reason about programs with two different sets of control operators. An obvious next step is to have level-n axiomatization for arbitrary $n > 0$. In the level-2 world, several things are different from the level-1 operators, for instance, the operator $\mathcal{A}$ cannot be regarded as a constant (value), and $\mathcal{C}$-lift no longer holds for $\mathcal{C}_2$. However, once we have obtained the level-2 axioms, it should be straightforward to formulate axioms of higher levels (just replace i by higher levels). The only problem we see is the telescope axiom, but Murthy [8] already gave a generalized telescope axiom for any level. In summary, going to level-3 or higher levels seems a straightforward extension of this paper (the details are yet to be done, though).

Besides going to higher levels, there are two major remaining problems: (1) the set of axioms we give in this paper may not be optimal in the sense that some axioms may be derived from others. In fact, for the case of level-1 axiomatization [6], we can derive $\mathcal{C}$-top and reset-reset from other level-1 axioms. At present, we do not know how to derive the level-2 instances of these axioms from other axioms. (2) we only worked for the $\mathcal{C}$ operators, while our actual aim is to axiomatize the $\mathcal{S}$ operator. It is easy just to obtain a sound and complete axiomatization for $\mathcal{S}$ using the translation given in this paper, but obtaining a *good* axiomatization needs more work. We believe this can be done in the same line as our previous work.

Acknowledgments: The author would like to thank Masahito Hasegawa who improved the axioms and the proof in the previous work, and also pointed out errors in earlier versions of this abstract. He also thanks to anonymous referees for constructive comments.

This work was supported in part by Grant-in-Aid for Scientific Research No. 13680411, from MEXT of Japan.

8. REFERENCES

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

[2] O. Danvy and Z. Yang. An Operational Investigation of the CPS Hierarchy. In *ESOP 99*, Lecture Notes in Computer Science 1576, pages 224–242, 1999.

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

[4] 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.

[5] 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.

[6] Y. Kameyama and M. Hasegawa. A Sound and Complete Axiomatization of Delimited Continuations. In *Proc. ACM International Conference on Functional Programming*, pages 177–188, 2003.

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

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

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

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

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