博士論文（要約）

論文題目　Meta-continuation Semantics
via Meta-lambda Calculus

(メタラムダ計算を用いたメタ継続意味論)

氏　　名　戸澤　一成

Delimited continuation is an abstract of continuation. Continuation is a notion representing "the rest of computation at some time," and using it as first-class objects enables us to represent kinds of control flows. By adding control delimiter we obtain a notion of *delimited continuation* that makes representation of control flows more effective and enables us to represent more computation effects. The call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus is introduced in order to characterize the expressibility of delimited continuation [AHS09]. It can operate delimited continuations as first-class objects. It is an extension of the call-by-value $\lambda\mu$-calculus. The difference from $\lambda\mu$-calculus is only the existence of a special continuation variable $\hat{\mathsf{tp}}$.

Since the variable $\hat{\mathsf{tp}}$ has the dynamic nature unlike the other variables, it is called a *dynamically-scoped variable*. The dynamic nature is explained as follows:

$$\begin{aligned}\textbf{let } x := (\lambda y.\mu\delta.[\alpha]y) \textbf{ in } \mu\alpha.[\alpha]x \to &\quad \mu\beta.[\beta](\lambda y.\mu\delta.[\alpha]y)\\ \textbf{let } x := (\lambda y.\mu\delta.[\hat{\mathsf{tp}}]y) \textbf{ in } \mu\hat{\mathsf{tp}}.[\hat{\mathsf{tp}}]x \to &\quad \mu\hat{\mathsf{tp}}.[\hat{\mathsf{tp}}](\lambda y.\mu\delta.[\hat{\mathsf{tp}}]y)\end{aligned}$$

Substitutions in the lambda calculus are performed in a capture-avoiding manner. On the other hand, Substitutions for the variable $\hat{\mathsf{tp}}$ contradict the variable-renaming rule. Therefore the binding structure about $\hat{\mathsf{tp}}$ is determined dynamically.

A CPS translation based on meta-continuation semantics is commonly used to give a semantics for the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus [DF90]. CPS translation is a method to give a semantics for a system, and its main role is to determine an evaluation strategy of the system. The CPS translation for the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus is seen as a two-fold translation, and therefore, it is called a *CPS2 translation*. The CPS2 translation has another role "to determine the dynamic nature of $\hat{\mathsf{tp}}$." Since the CPS2 translation plays the two roles simultaneously, it is difficult to consider the dynamic nature of delimited control separately. In order to discuss the dynamic nature separately from the call-by-value strategy, some kinds of decompositions of the CPS2 translation are proposed in previous studies, such as a decomposition of Kameyama for giving a CPS translation for higher-order delimited control [Kam07], one of Downen and Ariola for giving a semantics for an extension of $\lambda\mu\hat{\mathsf{tp}}$-calculus with multiple names of dynamically-scoped variables [DA12], and one of Ariola, Herbelin and Sabry for analyzing the structure of the type system for delimited control that is complicated because of control delimiter [AHS09].

The aim of this study is to separate the two roles of the CPS2 translation for the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus. In order to formulate it, we give two translations $\mathcal{C}$ and $\mathcal{D}$ whose composite is equal to the CPS2 translation. Each translation is designed to play only one role of the two roles respectively. In order to use each translation independently, we need to give a theory of the intermediate language of the decomposition.

We propose that *meta-lambda calculus* is useful as the intermediate system of the decomposition. Meta-lambda calculus is an extension of the lambda calculus with meta-variables and textual substitutions. Compared to usual substitutions,

textual substitution is executed as follows:

$$\begin{aligned}(\lambda x.\lambda y.x)(xy) \to &\quad \lambda z.xy\\ (\lambda X.\lambda y.X)(xy) \to &\quad \lambda y.xy\end{aligned}$$

While a variable renaming occurs during the substitution for the ordinary variable x, the substitution for the meta-variable X is executed textually. With respect to presence of renaming, textual substitutions are similar to substitutions for the dynamically-scoped variable $\hat{\mathsf{tp}}$. By observing this similarity, we construct a new translation $\mathcal{C}$. The translation $\mathcal{C}$ is an extension of the CPS translation for the call-by-value $\lambda\mu$-calculus with a meta-level structure in meta-lambda calculus. Therefore the translation $\mathcal{C}$ plays only the role of giving an evaluation strategy.

The key idea is to take a meta-lambda calculus supporting *cross-level computation* as the target of the translation $\mathcal{C}$. The translation $\mathcal{C}$ makes an unconventional use of meta-lambda calculus. Since variables at a higher level that occur in translated terms are regarded as object-level, we need to evaluate terms containing meta-variables, that is, we need the ability cross-level computation. Tobisawa introduced a meta-lambda calculus λ^* supporting cross-level computation [Tob15]. Cross-level computation enables us to define the translation $\mathcal{C}$ as an independent translation.

In Section 3, we define a target calculus $\lambda\hat{\mathsf{d}}^*$ of the translation $\mathcal{C}$. Since we need to consider cross-level computation of translated terms, $\lambda\hat{\mathsf{d}}^*$ is based on Tobisawa's meta-lambda calculus λ^*. In order to be a target of the translation $\mathcal{C}$, the meta-lambda calculus $\lambda\hat{\mathsf{d}}^*$ need to contain the ordinary extensional lambda calculus at the level-2 layer. Since λ^* does not have mechanisms for variable renaming and η-reduction, we give α-conversion and η-reduction for level-2 variables. Moreover, since $\lambda\hat{\mathsf{d}}^*$ need to have only a name $\hat{\mathsf{d}}$ for level-1 variables that is a counterpart of the dynamically-scoped variable $\hat{\mathsf{tp}}$, and since $\lambda\hat{\mathsf{d}}^*$ is too detailed for our purpose, we give a suitable class of terms to equate more terms than λ^*. At the end of Section 3, we show the confluence of $\lambda\hat{\mathsf{d}}^*$.

Proposition 1. *$\to_{\beta\eta}$ is confluent.*

In Section 4, we show that the translation $\mathcal{C}$ is sound and complete with respect to the CPS2 theory. The proof is based on the technique introduced in [SF92]. Since translated terms are independent of an evaluation order in the target calculus as the CPS2 translation, we can regard the translation $\mathcal{C}$ as a CPS translation whose target is a meta-lambda calculus. By the translation $\mathcal{C}$, the dynamic nature of $\hat{\mathsf{tp}}$ is derived naturally from the level structure on meta-lambda calculus. The results in Section 4 indicate that the translation $\mathcal{C}$ is another method to get the meta-continuation semantics, and gives a simpler understanding for the axioms regarding the dynamic nature of $\hat{\mathsf{tp}}$ by using mechanisms for dynamic binding in meta-lambda calculus.

Proposition 2. *The CPS2 theory is sound and complete with respect to the translation $\mathcal{C}$.*

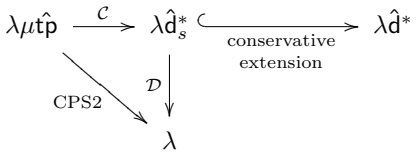

Figure 1: The relation to the CPS2 translation

In Section 5, we give a decomposition of the CPS2 translation into the translation $\mathcal{C}$, as shown in Figure 1. It is a modified decomposition of Downen and Ariola in the sense that we correct some errors in theirs and we give an axiomatization of the intermediate system. We define a translation $\mathcal{D}$ and show that the composite of the translations $\mathcal{C}$ and $\mathcal{D}$ is equal to the CPS2 translation. Moreover, we show that the translation $\mathcal{D}$ is sound and complete with respect to the equational theory $=_{\beta\eta}$ of the meta-lambda calculus $\lambda\hat{\mathsf{d}}^*$. It indicates that the decomposition separates the two roles of the CPS2 translation.

Proposition 3. *Let M be a term, and J a jump of the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus. Then the following hold:*

$$\begin{aligned} \mathcal{D}[\![\mathcal{C}[\![M]\!]_v]\!] &= [\![M]\!]_v \\ \mathcal{D}[\![\mathcal{C}[\![J]\!]_v]\!] &= [\![J]\!]_v \end{aligned}$$

Proposition 4. *The equational theory $=_s$ is sound and complete with respect to the translation $\mathcal{D}$.*

In Section 6, we establish a correspondence between the type systems of the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus and the meta-lambda calculus $\lambda\hat{\mathsf{d}}^*$. Danvy and Filinski's type system of the call-by-value $\lambda\mu\hat{\mathsf{tp}}$-calculus has parameterized structure [DF89]. Typing judgements in this system are of the form:

$$\Gamma\backslash S_1 \vdash M : T\backslash S_2|\Delta,$$

The difference from simple type theory is the existence of type parameters S_1 and S_2. Similarly, function types become of the form $[A\backslash S_1 \to B\backslash S_2]$. These parameters contribute the ability of each occurrence of control delimiter to have its type. Therefore the type system is said to support *answer-type modification*. On the other hand, it is known that modal type theory is useful for meta-lambda calculus. By using modal operators we can add information about the context of a term, so that the type system is called *contextual modal type theory* [NPP08]. The information enables us to prevent the textual substitutions that may raise a type mismatch error. In order to establish the correspondence between the type systems, we present a type theory of the meta-lambda calculus $\lambda\hat{\mathsf{d}}^*$ based on contextual modal type theory, and show the subject reduction property of this system. By defining properly a translation of types, the translation $\mathcal{C}$ is extended to the type systems so that parameterized structure is translated into

the modality in contextual modal type theory. We prove the type soundness of the translation $\mathcal{C}$, which means that the dynamic nature of $\hat{\mathsf{tp}}$ is explained in terms of contextual validity.

Proposition 5. *Suppose that*

$$\Delta; \hat{\mathsf{d}} : S \vdash M : T,$$

and suppose that $M \to_{\beta\eta} N$*. Then*

$$\Delta; \hat{\mathsf{d}} : S \vdash N : T.$$

Proposition 6. *Let* M *be a term and* J *a jump of the call-by-value* $\lambda\mu\hat{\mathsf{tp}}$*-calculus. Then the following hold:*

1. *If* $\Gamma \backslash S_1 \vdash M : A \backslash S_2 | \Delta$ *then* $\Gamma^*, \Delta^+; \hat{\mathsf{d}} : K_{S_1} \vdash \mathcal{C}[\![M]\!]_v : C_{A \backslash S_2}$.
2. *If* $\Gamma \backslash S_1 \vdash J : \perp\!\!\!\perp \mid \Delta$ *then* $\Gamma^*, \Delta^+; \hat{\mathsf{d}} : K_{S_1} \vdash \mathcal{C}[\![J]\!]_v : R$.

Bibliography

[AHS09] Zena M. Ariola, Hugo Herbelin, and Amr Sabry. A type-theoretic foundation of delimited continuations. *Higher-Order and Symbolic Computation*, 22(3):233–273, 2009.

[DA12] Paul Downen and Zena M. Ariola. A systematic approach to delimited control with multiple prompts. In *Proceedings of Programming Languages and Systems - 21st European Symposium on Programming (ESOP 2012)*, pages 234–253, 2012.

[DF89] Olivier Danvy and Andrzej Filinski. A functional abstraction of typed contexts. Technical Report DIKU-89/12, University of Copenhagen, 1989.

[DF90] Olivier Danvy and Andrzej Filinski. Abstracting control. In *Proceedings of the 1990 ACM Conference on LISP and Functional Programming (LFP '90)*, pages 151–160, 1990.

[Kam07] Yukiyoshi Kameyama. Axioms for control operators in the cps hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *ACM Transactions on Computational Logic*, 9(3):23:1–23:49, 2008.

[SF92] Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. In *Proceedings of the 1992 ACM conference on LISP and functional programming (LFP '92)*, pages 288–298, 1992.

[Tob15] Kazunori Tobisawa. A meta lambda calculus with cross-level computation. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2015)*, pages 383–393, 2015.