Axiomatizing Subtyped Delimited Continuations

Marek Materzok

University of Wrocław
Wrocław, Poland
marek.materzok@cs.uni.wroc.pl

Abstract

We present direct equational axiomatizations of the call-by-value lambda calculus with the control operators $shift_0$ and $reset_0$ that generalize Danvy and Filinski's *shift* and *reset* in that they allow for abstracting control beyond the top-most delimited continuation. We address an untyped version of the calculus as well as a typed version with effect subtyping. For each of the calculi we present a set of axioms that we prove sound and complete with respect to the corresponding CPS translation.

1998 ACM Subject Classification D.3.3 Language Constructs and Features

Keywords and phrases Delimited Continuations, Continuation Passing Style, Axiomatization

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.521

1 Introduction

Control operators for delimited continuations allow to alter the control flow of programs by capturing the current continuation as a first-class value, which can be activated later. The most well-known are *shift*/*reset*, introduced by Danvy and Filinski in [3]. They have many important applications, including representing monads, partial evaluation, mobile computing, linguistics and operating systems [8].

The $shift_0/reset_0$ control operators were first introduced besides the well-known *shift*/*reset* by Danvy and Filinski in [3]. The operators were recently found to have many desirable properties [8]. They can, like *shift*/*reset*, be described with a CPS translation. They have an interesting type system, which distinguishes between side-effect free and effectful terms. They can express the whole CPS hierarchy, in both typed and untyped settings [9]. And they recently helped to construct a theory of multiple prompts [4].

We are interested in the problem of reasoning directly about code using the $shift_0/reset_0$ control operators. Specifically, we look for a set of equational axioms which are sound and complete with respect to the CPS translation – and thus allow for the same reasoning which is possible on the CPS code. Previously, Sabry and Felleisen have given such axioms for call-by-value lambda calculus with *call/cc* [10][12]. Kameyama and Hasegawa solved the problem for *shift*/*reset* control operators [6]. Axioms for the CPS Hierarchy were given by Kameyama [5].

In this paper, we present the axiomatization for $shift_0/reset_0$ control operators, and prove soundness and completeness with respect to the CPS translation. We do this both in the untyped setting and in the typed setting with effect subtyping, where we use a type-directed selective CPS translation which takes subtyping into account. The proof method is a variation of the one presented by Sabry in [11]. Crucial for the proof is the \$ control operator, which was described and formalized in [9].

The paper is organized as follows. We introduce the $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$ languages and their CPS translations in Section 2. Our untyped axiomatizations for the two languages are given in

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 521–539

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$
\begin{array}{rcll}
\mathcal{C}[\![x]\!] & = & \lambda k.\, k\, x & \\
\mathcal{C}[\![\lambda x.\, e]\!] & = & \lambda k.\, k\, (\lambda x.\, \mathcal{C}[\![e]\!]) & \\
\mathcal{C}[\![e_1\, e_2]\!] & = & \lambda k.\, \mathcal{C}[\![e_1]\!]\, (\lambda f.\, \mathcal{C}[\![e_2]\!]\, (\lambda x.\, f\, x\, k)) & \\
\mathcal{C}[\![\mathcal{S}_0 k.\, e]\!] & = & \lambda k.\, \mathcal{C}[\![e]\!] & \\
\mathcal{C}[\![\langle e \rangle]\!] & = & \mathcal{C}[\![e]\!]\, (\lambda x.\, \lambda k.\, k\, x) & \textit{for } \lambda_{\mathcal{S}_0} \\
\mathcal{C}[\![e_1\, \$\, e_2]\!] & = & \lambda k.\, \mathcal{C}[\![e_1]\!]\, (\lambda f.\, \mathcal{C}[\![e_2]\!]\, f\, k) & \textit{for } \lambda_{\$}
\end{array}
$$

Figure 1 CPS translations for $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$.

Section 3. We describe the typed variants of the two languages, $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$, in Section 4. We present the typed axiomatizations in Section 5. In Section 6 we discuss the axioms and relate them to axioms for different systems of delimited continuations. Finally, we conclude in Section 7.

2 The languages $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$

Before we present the main results of the paper, we need to introduce and formally describe the languages used.

2.1 The language $\lambda_{\mathcal{S}_0}$ (*shift*$_0$/*reset*$_0$)

First, we define syntactic categories for expressions, values and evaluation contexts in the language $\lambda_{\mathcal{S}_0}$:

$$
e \;::=\; v \mid \mathcal{S}_0 x.\, e \mid e\, e \mid \langle e \rangle \qquad v \;::=\; x \mid \lambda x.\, e \qquad E \;::=\; \bullet \mid E\, e \mid v\, E
$$

The evaluation contexts are represented inside-out, which is formalized by the following definition of plugging a term inside a context:

$$
\bullet[e] \;=\; e \qquad (E\, e_2)[e] \;=\; E[e\, e_2] \qquad (v\, E)[e] \;=\; E[v\, e]
$$

The *shift*$_0$ operator $\mathcal{S}_0 x.\, \cdot$ captures the surrounding context up to (and including) the nearest dynamically surrounding delimiter $\langle \cdot \rangle$, which is then removed. The delimiter can also be removed when the enclosed expression is a value. This is displayed by the following reduction rules:

$$
\langle E[\mathcal{S}_0 x.\, e] \rangle \;\to\; e[\lambda x.\, \langle E[x] \rangle / x] \qquad \langle v \rangle \;\to\; v
$$

The language has also the standard beta and eta reductions. We will not concern ourselves more about the reduction rules, because the subject of this paper is the CPS semantics.

The CPS translation for the language $\lambda_{\mathcal{S}_0}$ is shown in Figure 1. (Please ignore for the moment the line marked "for $\lambda_{\$}$".) This is the untyped CPS translation first defined in [8] and proven correct with respect to the reduction rules.

The idea behind the translation is that the successive lambda abstractions (introduced by the translation of *shift*$_0$) bind continuations delimited by successive *reset*$_0$'s. It can be thought of as an infinitely-iterated (on the final answer position, like in the CPS Hierarchy) CPS translation, but eta reduced. In other words, we have potentially infinite number of continuation „levels", and *shift*$_0$/*reset*$_0$ operators allow us to change the current level.

2.2 The language $\lambda_{\$}$ (shift$_0$/\$)

The language $\lambda_{\$}$ was first defined in [9]. It is a generalization of $\lambda_{\mathcal{S}_0}$, a variant of which was explored by Kiselyov and Shan in [7]. The language plays a vital role in proving completeness of the axioms for $\lambda_{\mathcal{S}_0}$, and it is very interesting on its own. We describe it in this subsection.

As with $\lambda_{\mathcal{S}_0}$, we begin with introducing syntactic categories for expressions, values and evaluation contexts:

$$e \quad ::= \quad v \mid \mathcal{S}_0 x.\, e \mid e\, e \mid e \,\$\, e \qquad v \quad ::= \quad x \mid \lambda x.\, e \qquad E \quad ::= \quad \bullet \mid E\, e \mid v\, E \mid E \,\$\, e$$

We see that the *reset*$_0$ operator $\langle\cdot\rangle$ from $\lambda_{\mathcal{S}_0}$ was replaced by the (right-associative) binary operator \$. It is a generalization of *reset*$_0$: while the expression $\langle e\rangle$ means "evaluate e inside a new, empty context", $e_1 \,\$\, e_2$ means "evaluate e_2 inside a context terminated with e_1". We can express *reset*$_0$ using \$ by writing $(\lambda x.\, x) \,\$\, e$ instead of $\langle e\rangle$.

The \$ operator allows to easily restore captured contexts: the expression $\mathcal{S}_0 k.\, k \,\$\, e$ (where $k \notin \mathrm{V}(e)$) means the same as e. (We will make this statement formal in the following section.)

We define plugging terms inside evaluation contexts as follows:

$$\begin{array}{rclcrcl} \bullet[e] & = & e & & (v\, E)[e] & = & E[v\, e] \\ (E\, e_2)[e] & = & E[e\, e_2] & & (E \,\$\, e_2)[e] & = & E[e \,\$\, e_2] \end{array}$$

We have the following reduction rules for *shift*$_0$/\$, which generalize the reduction rules for *shift*$_0$/*reset*$_0$:

$$v \,\$\, E[\mathcal{S}_0 x.\, e] \quad \to \quad e[\lambda x.\, v \,\$\, E[x]/x] \qquad\qquad v_1 \,\$\, v_2 \quad \to \quad v_1\, v_2$$

Again, as with $\lambda_{\mathcal{S}_0}$, we give the meaning of $\lambda_{\$}$ terms with a CPS translation. It is shown in Figure 1.

The translation differs from the one for $\lambda_{\mathcal{S}_0}$ (shown in the same figure) only on the rule for the delimiter: while in the translation for *reset*$_0$ the translated subexpression has the (CPS-translated) identity function applied to it, in the translation for \$ the translated left subexpression is evaluated and applied to the translated right subexpression.

3 Untyped axiomatization

In this section we present sound and complete (with respect to the untyped translations of Figure 1) equational axiomatizations of $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$.

3.1 The axioms for $\lambda_{\mathcal{S}_0}$

The axioms for $\lambda_{\mathcal{S}_0}$ are presented in Figure 2. The first two (β_v and η_v) are the standard beta-value conversions. The third (β_Ω) is a beta-conversion restricted to evaluation contexts. The fourth and fifth ($\langle\mathcal{S}_0\rangle$ and $\langle v\rangle$) are equational versions of the reductions from the reduction semantics. These five axioms are standard and expected, the last two are interesting.

The axiom $\eta_{\langle\cdot\rangle}$ says that it is always possible to capture a continuation and restore it without changing the meaning of the expression. The axiom implies the existence of a potentially infinite tower of *reset*$_0$'s outside any expression. This is expected – the untyped CPS translation of Figure 1 is related to infinitely-iterated standard CPS translation, as discussed before in Section 2.1. The axiom is similar to Kameyama and Hasegawa's $\mathcal{S}$-elim; we discuss the connection in Section 6.1.

The last axiom, $\langle\lambda\rangle$, asserts that in expressions of the form $\langle(\lambda x.\, e_1)\, e_2\rangle$ we know that the topmost continuation for e_1 must be empty, and we can always throw it away and replace it with a new empty continuation.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & (\beta_v) \\
\lambda x.\, v\, x & = & v \quad x \notin \mathrm{V}(v) & (\eta_v) \\
(\lambda x.\, E[x])\, e & = & E[e] \quad x \notin \mathrm{V}(E) & (\beta_\Omega) \\
\langle E[\mathcal{S}_0 x.\, e]\rangle & = & e[\lambda x.\, \langle E[x]\rangle/x] \quad x \notin \mathrm{V}(E) & (\langle \mathcal{S}_0 \rangle) \\
\langle v\rangle & = & v & (\langle v \rangle) \\
\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e\rangle & = & e \quad k \notin \mathrm{V}(e) & (\eta_{\langle\cdot\rangle}) \\
\langle (\lambda x.\, \mathcal{S}_0 k.\, \langle e_1\rangle)\, e_2\rangle & = & \langle (\lambda x.\, e_1)\, e_2\rangle \quad k \notin \mathrm{V}(e_1) & (\langle\lambda\rangle)
\end{array}
$$

Figure 2 Axioms for $\lambda_{\mathcal{S}_0}$.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & (\beta_v) \\
\lambda x.\, v\, x & = & v \quad x \notin \mathrm{V}(v) & (\eta_v) \\
v \,\$\, \mathcal{S}_0 x.\, e & = & e[v/x] & (\beta_\$)
\end{array}
\qquad
\begin{array}{rcll}
\mathcal{S}_0 x.\, x \,\$\, e & = & e \quad x \notin \mathrm{V}(e) & (\eta_\$) \\
v_1 \,\$\, v_2 & = & v_1\, v_2 & (\$_v) \\
v \,\$\, E[e] & = & (\lambda x.\, v \,\$\, E[x]) \,\$\, e & (\$_E)
\end{array}
$$

Figure 3 Axioms for $\lambda_\$$.

3.2 The axioms for $\lambda_\$$

We present the axioms for $\lambda_\$$ in Figure 3. They are conceptually very different than the axioms for $\lambda_{\mathcal{S}_0}$, which may be surprising. But they are very regular and reveal the conceptual elegance of the $\lambda_\$$ language.

The first two axioms (β_v and η_v) are, as before, the standard beta-value conversions. The third one, $\beta_\$$, says that when we capture an empty context terminated with v, we get v back. The fourth, $\eta_\$$, means that we can always capture the top context, and then put it back as the terminating value on the delimiter. The two axioms can be thought of as beta and eta conversion axioms for *shift*$_0$ and $\$$ operators.

The fifth axiom, $\$_v$, says that if the inside of the context is a value, we can just apply it to the terminating value on the delimiter. This is an equational version of the reduction rule $v_1 \,\$\, v_2 \to v_1\, v_2$.

The last axiom, $\$_E$, says that we can move a suffix of the evaluation context delimited with $\$$ to the terminating value.

Please take notice that there is no axiom corresponding directly to the reduction rule $v \,\$\, E[\mathcal{S}_0 x.\, e] \to e[\lambda x.\, v \,\$\, E[x]/x]$. But the corresponding equation is still valid:

$$\lambda_\$ \vdash v \,\$\, E[\mathcal{S}_0 x.\, e] = (\lambda x.\, v \,\$\, E[x]) \,\$\, \mathcal{S}_0 x.\, e = e[\lambda x.\, v \,\$\, E[x]/x]$$

The β_Ω axiom also turned out to be redundant.

3.3 Reducing $\lambda_{\mathcal{S}_0}$ to $\lambda_\$$

In this subsection we show that the axioms for $\lambda_{\mathcal{S}_0}$ are sound and complete if and only if the axioms for $\lambda_\$$ are sound and complete. To achieve this, let us define a pair of translations – $\mathcal{D}[\![\cdot]\!]$ from $\lambda_{\mathcal{S}_0}$ to $\lambda_\$$, and $\mathcal{D}^{-1}[\![\cdot]\!]$ in the other direction:

$$
\begin{array}{rcl}
\mathcal{D}[\![\langle e\rangle]\!] & = & (\lambda x.\, x) \,\$\, \mathcal{D}[\![e]\!] \\
\mathcal{D}^{-1}[\![e_1 \,\$\, e_2]\!] & = & (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f\, x)\, \mathcal{D}^{-1}[\![e_2]\!]\rangle)\, \mathcal{D}^{-1}[\![e_1]\!]
\end{array}
$$

The remainder of the translations is defined homomorphically.

The translations have the following properties (λ consists of full β and η axioms):

$$
\begin{array}{rcl@{\qquad}rcl}
\mathcal{G}[\![x]\!] & = & \mathcal{S}_0 k.\, k\, x & \mathcal{P}[\![x]\!] & = & x \\
\mathcal{G}[\![\lambda x.\, e]\!] & = & \mathcal{S}_0 k.\, k\, (\lambda x.\, \mathcal{G}[\![e]\!]) & \mathcal{P}[\![\lambda x.\, e]\!] & = & \lambda x.\, \mathcal{P}[\![e]\!] \\
\mathcal{G}[\![e_1\, e_2]\!] & = & \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] & \mathcal{P}[\![v_1\, v_2]\!] & = & \mathcal{P}[\![v_1]\!]\, \mathcal{P}[\![v_2]\!] \\
\mathcal{G}[\![\mathcal{S}_0 k.\, e]\!] & = & \mathcal{S}_0 k.\, \mathcal{G}[\![e]\!] & \mathcal{P}[\![\mathcal{S}_0 x.\, e]\!] & = & \lambda x.\, \mathcal{P}[\![e]\!] \\
\mathcal{G}[\![e_1 \,\$\, e_2]\!] & = & \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] & \mathcal{P}[\![v \,\$\, e]\!] & = & \mathcal{P}[\![e]\!]\, \mathcal{P}[\![v]\!]
\end{array}
$$

Figure 4 CGS translation of $\lambda_\$$ to $\lambda_\$^{\mathrm{G}}$; translation of $\lambda_\$^{\mathrm{G}}$ to λ.

▶ Property 1. We have the following:

1. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] = e$.
2. For every $\lambda_\$$ term e we have $\lambda_\$ \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e]\!]]\!] = e$.
3. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]$.
4. For every $\lambda_\$$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e]\!]]\!]$.
5. $\lambda_{\mathcal{S}_0} \vdash e_1 = e_2$ implies $\lambda_\$ \vdash \mathcal{D}[\![e_1]\!] = \mathcal{D}[\![e_2]\!]$.
6. $\lambda_\$ \vdash e_1 = e_2$ implies $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$.

▶ **Theorem 2.** *The axioms for $\lambda_{\mathcal{S}_0}$ are sound iff the axioms for $\lambda_\$$ are sound.*

Proof. Suppose that the axioms for $\lambda_{\mathcal{S}_0}$ are sound. Assume $\lambda_\$ \vdash e_1 = e_2$. By Property 1.6 we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$. Using the assumed soundness of $\lambda_{\mathcal{S}_0}$ axioms gives $\lambda \vdash \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. From Property 1.4 we get $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$. The other direction is analogous. ◀

▶ **Theorem 3.** *The axioms for $\lambda_{\mathcal{S}_0}$ are complete iff the axioms for $\lambda_{\mathcal{S}_0}$ are complete.*

Proof. Suppose that the axioms for $\lambda_{\mathcal{S}_0}$ are complete. Assume $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$. From Property 1.4 we get $\lambda \vdash \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. Using the assumed completeness of $\lambda_{\mathcal{S}_0}$ axioms we get $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$. By Property 1.6 we have $\lambda_\$ \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] = \mathcal{D}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]$. From Property 1.2 follows the thesis. The other direction is analogous. ◀

3.4 CGS translation

Following the approach of Sabry [11], we show soundness and completeness of $\lambda_\$$ axioms in two steps. We introduce a translation from $\lambda_\$$ targeting a certain syntactical subset of $\lambda_\$$, which we call $\lambda_\$^{\mathrm{G}}$. We first prove soundness and completeness of $\lambda_\$$ with respect to $\lambda_\$^{\mathrm{G}}$, and then of $\lambda_\$^{\mathrm{G}}$ with respect to λ.

The language $\lambda_\$^{\mathrm{G}}$ is defined as follows:

$$e \quad ::= \quad \mathcal{S}_0 x.\, e \mid v\, v \mid v \,\$\, e \qquad v \quad ::= \quad x \mid \lambda x.\, e$$

In other words, we only allow applications with values on both sides and \$ with a value on the left side. Please also notice that the syntactic categories of expressions and values are separate in $\lambda_\$^{\mathrm{G}}$.

The translation is described in Figure 4. It is very similar to the CPS translation shown in Figure 1. The difference is that in the translation introduced in this section we use *shift*$_0$ and \$ instead of function abstraction and application for passing the continuation. We call this translation continuation-grabbing style (CGS) translation, because (as in the Sabry's translation) the terms actively „grab" their surrounding continuation, instead of passively waiting for it using a lambda abstraction, as is the case in the CPS translation.

The translation has an important property that by replacing shift_0 by λ and \$ by function application in target terms (Figure 4), we obtain the CPS translation from Figure 1:

▶ Property 4 (CPS-translation). $\mathcal{C}[\![e]\!] = \mathcal{P}[\![\mathcal{G}[\![e]\!]]\!]$

CGS terms are closed on β, η, $\beta_\$$ and $\eta_\$$ reductions. The equalities generated by these four reductions, restricted so that one cannot obtain non-CGS terms by expansion, form an axiomatization of $\lambda^{\mathrm{G}}_\$$.

We can easily prove soundness of $\lambda_\$$ axioms with respect to $\lambda^{\mathrm{G}}_\$$:

▶ **Lemma 5.** *If* $\lambda_\$ \vdash e_1 = e_2$*, then* $\lambda^G_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$.

In order to prove completeness, we need another important property of the CGS translation – that the target terms are equal in $\lambda_\$$ to the source terms:

▶ **Lemma 6.** *For every* $\lambda_\$$ *term* e *we have* $\lambda_\$ \vdash e = \mathcal{G}[\![e]\!]$.

We also make the observation that every pair of $\lambda^{\mathrm{G}}_\$$ terms equal in $\lambda^{\mathrm{G}}_\$$ is also equal in $\lambda_\$$:

▶ **Lemma 7.** *If* $\lambda^G_\$ \vdash e_1 = e_2$*, then* $\lambda_\$ \vdash e_1 = e_2$.

We can now easily prove completeness of $\lambda_\$$ axioms with respect to $\lambda^{\mathrm{G}}_\$$:

▶ **Lemma 8.** *If* $\lambda^G_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$*, then* $\lambda_\$ \vdash e_1 = e_2$.

Proof. Assume that $\lambda^{\mathrm{G}}_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$. By Lemma 7 we have $\lambda_\$ \vdash \mathcal{G}[\![e_1]\!] = \mathcal{G}[\![e_2]\!]$. The thesis follows from Lemma 6. ◀

3.5 From $\lambda^{\mathrm{G}}_\$$ to λ

Soundness of $\lambda^{\mathrm{G}}_\$$ axioms with respect to λ is trivial:

▶ **Lemma 9.** *If* $\lambda^G_\$ \vdash e_1 = e_2$*, then* $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$.

We still need to prove completeness of $\lambda^{\mathrm{G}}_\$$ axioms with respect to λ. This seems to be an easy task, but there is one important complication. Take a look at the translation in Figure 4. The translation replaces the \$ operator with function applications and the shift_0 operator with lambda abstractions. This causes new redexes to appear in the image of $\mathcal{P}[\![\cdot]\!]$; for example,

$$\lambda \vdash \mathcal{P}[\![\lambda x.\, x \,\$\, e]\!] = \lambda x.\, \mathcal{P}[\![e]\!]\, x = \mathcal{P}[\![e]\!]$$

But the $\lambda^{\mathrm{G}}_\$$ terms $\lambda x.\, x \,\$\, e$ and e are in separate syntactical categories – the one is a value, the other is an expression, and in $\lambda^{\mathrm{G}}_\$$ values are not expressions.

We introduce an intermediate language, $\lambda^{\mathrm{I}}_\$$, defined as follows:

$$e \quad ::= \quad \mathcal{S}_0 x.\, e \mid v\, v \mid v \,\$\, e \mid i_v[v] \qquad v \quad ::= \quad x \mid \lambda x.\, e \mid i_e[e]$$

The language is a syntactic extension of $\lambda^{\mathrm{G}}_\$$, which additionally allows using an expression as a value (and vice versa) with an explicit injection. Then we define two translations – $\mathcal{P}_I[\![\cdot]\!]$ from $\lambda^{\mathrm{I}}_\$$ to λ, and $\mathcal{I}[\![\cdot]\!]$ from $\lambda^{\mathrm{I}}_\$$ to $\lambda^{\mathrm{G}}_\$$:

$$\begin{array}{lcllcl} \mathcal{P}_I[\![i_v[v]]\!] & = & \mathcal{P}_I[\![v]\!] & \mathcal{I}[\![i_v[v]]\!] & = & \mathcal{S}_0 k.\, \mathcal{I}[\![v]\!]\, k \\ \mathcal{P}_I[\![i_e[e]]\!] & = & \mathcal{P}_I[\![e]\!] & \mathcal{I}[\![i_e[e]]\!] & = & \lambda x.\, x \,\$\, \mathcal{I}[\![e]\!] \end{array}$$

The translation $\mathcal{P}_I[\![\cdot]\!]$ is based on $\mathcal{P}[\![\cdot]\!]$, but ignores the explicit injections. The other translation, $\mathcal{I}[\![\cdot]\!]$, leaves most of the term unchanged (the cases not mentioned are defined homomorphically), but expands the injections so that the result is a valid $\lambda^{\mathrm{G}}_\$$ term. The expansions have the property that their translations to λ can be eta-reduced. Thus we have the following:

$$\frac{\tau \le \tau' \qquad \sigma \le \sigma'}{\tau\ \sigma \le \tau'\ \sigma'} \qquad \frac{}{\alpha \le \alpha} \qquad \frac{\tau_1' \le \tau_1 \qquad \tau_2\ \sigma \le \tau_2'\ \sigma'}{\tau_1 \xrightarrow{\sigma} \tau_2 \le \tau_1' \xrightarrow{\sigma'} \tau_2'}$$

$$\frac{}{\epsilon \le \epsilon} \qquad \frac{\tau_1\ \sigma_1 \le \tau_2\ \sigma_2}{\epsilon \le [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2} \qquad \frac{\tau_1'\ \sigma_1' \le \tau_1\ \sigma_1 \qquad \tau_2\ \sigma_2 \le \tau_2'\ \sigma_2'}{[\tau_1\ \sigma_1]\ \tau_2\ \sigma_2 \le [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2'}$$

$$\frac{}{\Gamma, x : \tau_1 \vdash x : \tau_1}\ \text{VAR} \qquad \frac{\Gamma \vdash e : \tau\ \sigma \qquad \tau\ \sigma \le \tau'\ \sigma'}{\Gamma \vdash e : \tau'\ \sigma'}\ \text{SUB} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2\ \sigma}{\Gamma \vdash \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2}\ \text{ABS}$$

$$\frac{\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2 \vdash e : \tau_3\ \sigma'}{\Gamma \vdash \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}\ \text{SFT} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2\ \sigma}\ \text{PAPP}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{[\tau_4'\ \sigma_4']\ \tau_3'\ \sigma_3'} \tau_2\ [\tau_2'\ \sigma_2']\ \tau_1'\ \sigma_1' \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_3'\ \sigma_3']\ \tau_2'\ \sigma_2'}{\Gamma \vdash e_1\ e_2 : \tau_2\ [\tau_4'\ \sigma_4']\ \tau_1'\ \sigma_1'}\ \text{APP}$$

Rule for $\lambda_{\mathcal{S}_0}$:

$$\frac{\Gamma \vdash e : \tau'\ [\tau']\ \tau\ \sigma}{\Gamma \vdash \langle e \rangle : \tau\ \sigma}\ \text{RST}$$

Rules for $\lambda_{\$}$:

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}{\Gamma \vdash e_1\ \$\ e_2 : \tau_3\ \sigma'}\ \text{PDOL}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \xrightarrow{\sigma} \tau_2\ [\tau_2'\ \sigma_2']\ \tau_1'\ \sigma_1' \qquad \Gamma \vdash e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ [\tau_3'\ \sigma_3']\ \tau_2'\ \sigma_2'}{\Gamma \vdash e_1\ \$\ e_2 : \tau_3\ [\tau_3'\ \sigma_3']\ \tau_1'\ \sigma_1'}\ \text{DOL}$$

Figure 5 The type systems $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$ (with subtyping).

▶ Property 10. For any $\lambda^{\mathrm{I}}_{\$}$ term e we have $\lambda \vdash \mathcal{P}_I[\![e]\!] = \mathcal{P}[\![\mathcal{I}[\![e]\!]]\!]$.

▶ Property 11. For any $\lambda^{\mathrm{G}}_{\$}$ term e we have $\mathcal{P}_I[\![e]\!] = \mathcal{P}[\![e]\!]$ and $\mathcal{I}[\![e]\!] = e$.

The equational theory for $\lambda^{\mathrm{I}}_{\$}$ consists of β_v, η_v, $\beta_{\$}$, $\eta_{\$}$ and the following two equalities:

$$i_v[v] \quad = \quad \mathcal{S}_0 x.\, v\, x \qquad (i_v) \qquad i_e[e] \quad = \quad \lambda x.\, x\, \$\, e \qquad (i_e)$$

The following lemma is trivial:

▶ **Lemma 12.** *For every two $\lambda^I_{\$}$ terms e_1 and e_2, if $\lambda^I_{\$} \vdash e_1 = e_2$, then $\lambda^G_{\$} \vdash \mathcal{I}[\![e_1]\!] = \mathcal{I}[\![e_2]\!]$.*

Thanks to the injections, we can prove the completeness lemma for $\lambda^{\mathrm{I}}_{\$}$:

▶ **Lemma 13.** *For every two $\lambda^I_{\$}$ expressions e_1 and e_2, if $\lambda \vdash \mathcal{P}_I[\![e_1]\!] = \mathcal{P}_I[\![e_2]\!]$, then $\lambda^I_{\$} \vdash e_1 = e_2$. The same holds for values.*

We can use it to prove completeness for $\lambda^{\mathrm{G}}_{\$}$:

▶ **Lemma 14.** *For every two $\lambda^G_{\$}$ terms e_1 and e_2, if $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, then $\lambda^G_{\$} \vdash e_1 = e_2$.*

Proof. Suppose that $\lambda \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$. By Property 11, we have $\lambda \vdash \mathcal{P}_I[\![e_1]\!] = \mathcal{P}_I[\![e_2]\!]$. Using Lemma 13 we get $\lambda^{\mathrm{I}}_{\$} \vdash e_1 = e_2$, Lemma 12 gives us $\lambda^{\mathrm{G}}_{\$} \vdash \mathcal{I}[\![e_1]\!] = \mathcal{I}[\![e_2]\!]$. By Property 11, we have $\lambda^{\mathrm{G}}_{\$} \vdash e_1 = e_2$. ◀

We can now finally prove soundness and completeness of the $\lambda_{\$}$ axioms:

▶ **Theorem 15** (Soundness). *If $\lambda_{\$} \vdash e_1 = e_2$, then $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$.*

▶ **Theorem 16** (Completeness). *If $\lambda \vdash \mathcal{C}[\![e_1]\!] = \mathcal{C}[\![e_2]\!]$, then $\lambda_{\$} \vdash e_1 = e_2$.*

$$
\begin{array}{rcll}
\mathcal{C}[\![e]\!]_{\mathrm{SUB}(\tau\ \sigma \le \tau'\ \sigma', D)} & = & \mathcal{C}[\![\tau\ \sigma \le \tau'\ \sigma']\!][\mathcal{C}[\![e]\!]_D] & \\
\mathcal{C}[\![x]\!]_{\mathrm{VAR}} & = & x & \\
\mathcal{C}[\![\lambda x.\, e]\!]_{\mathrm{ABS}(D)} & = & \lambda x.\, \mathcal{C}[\![e]\!]_D & \\
\mathcal{C}[\![e_1\ e_2]\!]_{\mathrm{PAPP}(D_1, D_2)} & = & \mathcal{C}[\![e_1]\!]_{D_1}\ \mathcal{C}[\![e_2]\!]_{D_2} & \\
\mathcal{C}[\![e_1\ e_2]\!]_{\mathrm{APP}(D_1, D_2)} & = & \lambda k.\, \mathcal{C}[\![e_1]\!]_{D_1}\ (\lambda f.\, \mathcal{C}[\![e_2]\!]_{D_2}\ (\lambda x.\, f\ x\ k)) & \\
\mathcal{C}[\![\mathcal{S}_0 x.\, e]\!]_{\mathrm{SFT}(D)} & = & \lambda x.\, \mathcal{C}[\![e]\!]_D & \\
\mathcal{C}[\![e_1\ \$\ e_2]\!]_{\mathrm{PDOL}(D_1, D_2)} & = & \mathcal{C}[\![e_2]\!]_{D_2}\ \mathcal{C}[\![e_1]\!]_{D_1} & \text{for } \lambda^{\le}_{\$} \vee \lambda^{\le}_{\mathcal{S}_0} \\
\mathcal{C}[\![e_1\ \$\ e_2]\!]_{\mathrm{DOL}(D_1, D_2)} & = & \lambda k.\, \mathcal{C}[\![e_1]\!]_{D_1}\ (\lambda f.\, \mathcal{C}[\![e_2]\!]_{D_2}\ f\ k) & \text{for } \lambda^{\le}_{\$} \vee \lambda^{\le}_{\mathcal{S}_0} \\
\mathcal{C}[\![\langle e \rangle]\!]_{\mathrm{RST}(D)} & = & \mathcal{C}[\![e]\!]_D\ (\lambda x.\, x) & \text{for } \lambda^{\le}_{\mathcal{S}_0}
\end{array}
$$

$$
\begin{array}{rcl}
\mathcal{C}[\![\alpha \le \alpha]\!][e] & = & e \\
\mathcal{C}[\![\tau_1' \xrightarrow{\sigma_1} \tau_1 \le \tau_2' \xrightarrow{\sigma_2} \tau_2]\!][e] & = & \lambda x.\, \mathcal{C}[\![\tau_1\ \sigma_1 \le \tau_2\ \sigma_2]\!][e\, \mathcal{C}[\![\tau_2' \le \tau_1']\!][x]] \\
\mathcal{C}[\![\tau_1\ \epsilon \le \tau_2\ \epsilon]\!][e] & = & \mathcal{C}[\![\tau_1 \le \tau_2]\!][e] \\
\mathcal{C}[\![\tau\ \epsilon \le \tau'\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!][e] & = & \lambda k.\, \mathcal{C}[\![\tau_1\ \sigma_1 \le \tau_2\ \sigma_2]\!][k\, \mathcal{C}[\![\tau \le \tau']\!][e]] \\
\mathcal{C}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2 \le \tau'\ [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2']\!][e] & = & \lambda k.\, \mathcal{C}[\![\tau_1'\ \sigma_1' \le \tau_2'\ \sigma_2']\!][e \\
 & & \quad (\lambda x.\, \mathcal{C}[\![\tau_1'\ \sigma_1' \le \tau_1\ \sigma_1]\!][k\, \mathcal{C}[\![\tau \le \tau']\!][x]])]
\end{array}
$$

Figure 6 Type-directed selective CPS translations for $\lambda^{\le}_{\mathcal{S}_0}$ to $\lambda^{\le}_{\$}$.

4 Typed languages $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$

We now take a break from equational axiomatizations and describe typed versions of $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$, called $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$. We present sound and complete axiomatizations for these languages in the next section.

In this work we use explicit type annotations on bound variables. This style of presentation of typed languages is called „de Bruijn style" by Barendregt [1]. Thus the syntax of the $\lambda^{\le}_{\mathcal{S}_0}$ and $\lambda^{\le}_{\$}$ languages is the same as for $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$, with the following changes (τ is the syntactic category of types):

$$e \quad ::= \quad v \mid \mathcal{S}_0 x : \tau.\, e \mid \ldots \qquad v \quad ::= \quad x \mid \lambda x : \tau.\, e$$

We often omit the type annotations for clarity, but they are still implicitly present.

4.1 Type systems

The description is shortened because of space limitations; for more details, see [8] and [9]. First let us define syntactic categories of types and effect annotations:

$$\tau \quad ::= \quad \alpha \mid \tau \xrightarrow{\sigma} \tau \qquad \sigma \quad ::= \quad \epsilon \mid [\tau\ \sigma]\ \tau\ \sigma$$

An effect annotation can only be given meaning together with the type it annotates. The typing judgment $\Gamma \vdash e : \tau_1'\ [\tau_1\ \sigma_1]\ \ldots \tau_n'\ [\tau_n\ \sigma_n]\ \tau\ \epsilon$ means "the expression e, when evaluated inside contexts of types $\tau_1' \xrightarrow{\sigma_1} \tau_1, \ldots, \tau_n' \xrightarrow{\sigma_n} \tau_n$, gives an answer of type τ. In particular, the judgment $\Gamma \vdash e : \tau\ \epsilon$ means "the expression e has no control effects and, when evaluated, yields a value of type τ". We will often omit ϵ where it leads to no confusion.

The type systems are shown in Figure 5. The type system for $\lambda^{\le}_{\$}$ is the one presented in [9]. The type system for $\lambda^{\le}_{\mathcal{S}_0}$ differs slightly only in the rule PAPP from the one from [8]. The modification does not change the expressiveness of the type system, but helps with the proofs.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, p & = & e[p/x] & (\beta_p) \\
\lambda x.\, p\, x & = & p & x \notin \mathrm{V}(p) \quad (\eta_p) \\
(\lambda x.\, E[x])\, e & = & E[e] & x \notin \mathrm{V}(E) \quad (\beta_\Omega) \\
\langle E[\mathcal{S}_0 x.\, e] \rangle & = & e[\lambda x.\, \langle E[x] \rangle / x] & x \notin \mathrm{V}(E) \quad (\langle \mathcal{S}_0 \rangle) \\
\langle p \rangle & = & p & (\langle p \rangle) \\
\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e \rangle & = & e & k \notin \mathrm{V}(e) \quad (\eta_{\langle\cdot\rangle}) \\
\langle (\lambda x.\, \mathcal{S}_0 k.\, \langle e_1 \rangle)\, e_2 \rangle & = & \langle (\lambda x.\, e_1)\, e_2 \rangle & k \notin \mathrm{V}(e_1) \quad (\langle\lambda\rangle)
\end{array}
$$

Figure 7 Axioms for $\lambda^{\leq}_{\mathcal{S}_0}$.

$$
\begin{array}{rcll}
(\lambda x.\, e)\, p & = & e[p/x] & (\beta_p) \\
\lambda x.\, p\, x & = & p \quad x \notin \mathrm{V}(p) & (\eta_p) \\
p \,\$\, \mathcal{S}_0 x.\, e & = & e[p/x] & (\beta^p_\$)
\end{array}
\qquad
\begin{array}{rcll}
\mathcal{S}_0 x.\, x \,\$\, e & = & e \quad x \notin \mathrm{V}(e) & (\eta_\$) \\
p_1 \,\$\, p_2 & = & p_1\, p_2 & (\$_p) \\
p \,\$\, E[e] & = & (\lambda x.\, p \,\$\, E[x]) \,\$\, e & (\$_E)
\end{array}
$$

Figure 8 Axioms for $\lambda^{\leq}_{\$}$.

The type systems include three subtyping relations, defined on types, effect annotations and annotated types, which are also defined in Figure 5. The subtyping relations are partial orders: they are reflexive, weakly antisymmetric and transitive. We also have the following:

▶ Property 17. Every derivable subtyping judgment has only one derivation.

The property allows us to identify a subtyping judgment with its only derivation, which is important for our proofs.

4.2 Selective CPS translations

For the typed languages $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$ we can define different translations than these defined in Figure 1. The type information can be used to preserve pure (or control effect free) code without changes and CPS-translate only the impure parts. The translations are shown in Figure 6. The translation for $\lambda^{\leq}_{\mathcal{S}_0}$ is the one from [8]; the one for $\lambda^{\leq}_{\$}$ is derived from it.

These translations preserve types in the following sense. Let us define translations from $\lambda^{\leq}_{\mathcal{S}_0}$ types and typed annotations to simple types of $\lambda^{\to}$:

$$
\begin{array}{rcl}
\mathcal{C}[\![\alpha]\!] & = & \alpha \\
\mathcal{C}[\![\tau' \xrightarrow{\sigma} \tau]\!] & = & \mathcal{C}[\![\tau']\!] \to \mathcal{C}[\![\tau\ \sigma]\!]
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{C}[\![\tau\ \epsilon]\!] & = & \mathcal{C}[\![\tau]\!] \\
\mathcal{C}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!] & = & (\mathcal{C}[\![\tau]\!] \to \mathcal{C}[\![\tau_1\ \sigma_1]\!]) \to \mathcal{C}[\![\tau_2\ \sigma_2]\!]
\end{array}
$$

We have the following:

▶ Property 18 (Type preservation). If D is a derivation of $\Gamma \vdash e : \tau\ \sigma$ in $\lambda^{\leq}_{\mathcal{S}_0}$, then $\mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e]\!]_D : \mathcal{C}[\![\tau\ \sigma]\!]$ in $\lambda^{\to}$. This also holds for $\lambda^{\leq}_{\$}$.

5 Typed axiomatization

In this section we present sound and complete (with respect to the type-directed selective translation of Figure 6) equational axiomatizations of $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$. The development mostly follows the untyped one, but there are a few surprises, starting with the axioms themselves.

5.1 The typed axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$

We present the axioms in Figure 7 and Figure 8. We use the letter p to denote pure expressions (the ones which can be typed with the empty effect annotation ϵ). The axioms seem similar to the untyped ones, but several things need to be noted.

First, because the axioms themselves are typed, they can only be used when the types match. For example, the typed $\eta_\$$ axiom cannot be typed pure, so it is only applicable on terms with an impure type. (Therefore the implicit infinite tower of resets, which is present in the untyped languages, disappears in typed ones.)

Second, the dependence on types makes it important to mention the typing context, which gives types to variables. Because the subtyping rule allows the same term to have different types, the concrete type which we consider needs also to be mentioned. Thus we use the notation $\lambda^{\leq}_{\mathcal{S}_0}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$ when talking about equality modulo the axioms.

Third, in the typed axioms the syntactical value restriction present in the untyped axioms β_v, η_v, $\beta_\$$, $\langle v \rangle$ and $\$_v$ is replaced by type-dependent purity restriction. This change is caused by the fact that the CPS translations considered are selective – they leave the pure terms unchanged. The typed axioms are more general, because every value has a pure typing.

Finally, we point out that the axioms give a call-by-name interpretation to pure subprograms. This is not problematic because the language considered is terminating and has no side effects other than capturing of delimited contexts by shift_0.

5.2 Reducing $\lambda^{\leq}_{\mathcal{S}_0}$ to $\lambda^{\leq}_{\$}$

Similar to the untyped case, the axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$ are related by the following theorem. We omit the proof because of space limitations.

▶ **Theorem 19.** *The axioms for $\lambda^{\leq}_{\mathcal{S}_0}$ are sound (complete) iff the axioms for $\lambda^{\leq}_{\$}$ are sound (complete).*

5.3 Typed CGS translation

Analogously to the untyped case, we present a translation from $\lambda^{\leq}_{\$}$ which targets a certain subset of it, called $\lambda^{\rightarrow}_{\$}$. Differently to the untyped case, we will define this subset not by restricting syntax, but by using a simpler type system, which consists of rules VAR, ABS, SFT, PAPP and PDOL (Figure 5).

It is worth notice that the restrictions imposed by the restricted type system for $\lambda^{\rightarrow}_{\$}$ are analogous to the syntactic restrictions on $\lambda^{\mathrm{G}}_{\$}$: the restriction of being a value in the untyped case corresponds to the restriction of having a pure typing in the typed case. Another interesting point is that there is no subtyping in the type system. The rules for impure application and impure \$ are also gone, and with good reason: without subtyping, these rules fail subject reduction.

The axioms β_p, η_p, $\beta^p_\$$ and $\eta_\$$ form an axiomatization of $\lambda^{\rightarrow}_{\$}$. Take notice that the type system of $\lambda^{\rightarrow}_{\$}$ makes the full beta reduction valid, because it forces the type of the function argument to be pure. Therefore, as in the untyped case, the typed CGS language is evaluation order independent.

We present the typed CGS translation in Figure 9. The translation is derived from the typed CPS translation in Figure 6 using the same principles as with the untyped one. As before, replacing the occurrences of shift_0 with lambda abstractions and occurrences of \$ with function applications (as in Figure 4, but extended to work on terms with type annotations). in the result terms of $\mathcal{G}[\![\cdot]\!]$. gives us the CPS translation:

$$
\begin{aligned}
\mathcal{G}[\![e]\!]_{\mathrm{SUB}(\tau\ \sigma\leq\tau'\ \sigma',D)} &= \mathcal{G}[\![\tau\ \sigma\leq\tau'\ \sigma']\!][\mathcal{G}[\![e]\!]_D]\\
\mathcal{G}[\![x]\!]_{\mathrm{VAR}} &= x\\
\mathcal{G}[\![\lambda x.\,e]\!]_{\mathrm{ABS}(D)} &= \lambda x.\,\mathcal{G}[\![e]\!]_D\\
\mathcal{G}[\![e_1\ e_2]\!]_{\mathrm{PAPP}(D_1,D_2)} &= \mathcal{G}[\![e_1]\!]_{D_1}\ \mathcal{G}[\![e_2]\!]_{D_2}\\
\mathcal{G}[\![e_1\ e_2]\!]_{\mathrm{APP}(D_1,D_2)} &= \mathcal{S}_0k.\,(\lambda f.\,(\lambda x.\,k\,\$\,f\,x)\,\$\,\mathcal{G}[\![e_2]\!]_{D_2})\,\$\,\mathcal{G}[\![e_1]\!]_{D_1}\\
\mathcal{G}[\![\mathcal{S}_0x.\,e]\!]_{\mathrm{SFT}(D)} &= \mathcal{S}_0x.\,\mathcal{G}[\![e]\!]_D\\
\mathcal{G}[\![e_1\,\$\,e_2]\!]_{\mathrm{PDOL}(D_1,D_2)} &= \mathcal{G}[\![e_1]\!]_{D_1}\,\$\,\mathcal{G}[\![e_2]\!]_{D_2}\\
\mathcal{G}[\![e_1\,\$\,e_2]\!]_{\mathrm{DOL}(D_1,D_2)} &= \mathcal{S}_0k.\,(\lambda f.\,k\,\$\,f\,\$\,\mathcal{G}[\![e_2]\!]_{D_2})\,\$\,\mathcal{G}[\![e_1]\!]_{D_1}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{G}[\![\alpha\leq\alpha]\!][e] &= e\\
\mathcal{G}[\![\tau_1'\xrightarrow{\sigma_1}\tau_1\leq\tau_2'\xrightarrow{\sigma_2}\tau_2]\!][e] &= \lambda x.\,\mathcal{G}[\![\tau_1\ \sigma_1\leq\tau_2\ \sigma_2]\!][e\,\mathcal{G}[\![\tau_2'\leq\tau_1']\!][x]]\\
\mathcal{G}[\![\tau_1\ \epsilon\leq\tau_2\ \epsilon]\!][e] &= \mathcal{G}[\![\tau_1\leq\tau_2]\!][e]\\
\mathcal{G}[\![\tau\ \epsilon\leq\tau'\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2]\!][e] &= \mathcal{S}_0k.\,\mathcal{G}[\![\tau_1\ \sigma_1\leq\tau_2\ \sigma_2]\!][k\,\mathcal{G}[\![\tau\leq\tau']\!][e]]\\
\mathcal{G}[\![\tau\ [\tau_1\ \sigma_1]\ \tau_2\ \sigma_2\leq\tau'\ [\tau_1'\ \sigma_1']\ \tau_2'\ \sigma_2']\!][e] &= \mathcal{S}_0k.\,\mathcal{G}[\![\tau_1'\ \sigma_1'\leq\tau_2'\ \sigma_2']\!][\\
&\qquad(\lambda x.\,\mathcal{G}[\![\tau_1'\ \sigma_1'\leq\tau_1\ \sigma_1]\!][k\,\mathcal{G}[\![\tau\leq\tau']\!][x]])\,\$\,e]
\end{aligned}
$$

Figure 9 Type-directed selective CGS translation of $\lambda^{\leq}_{\$}$ to $\lambda^{\rightarrow}_{\$}$.

▶ Property 20 (CPS translation). $\mathcal{C}[\![e]\!]_D = \mathcal{P}[\![\mathcal{G}[\![e]\!]_D]\!]$

In contrast to the untyped case, the soundness of $\lambda^{\leq}_{\$}$ axioms is not trivial. The reason is that the typed CGS (and CPS) translation depends on the typing derivation. It is easy to show that every axiom is sound in some particular derivation, but we need to have them sound in any derivation to have soundness. Therefore, we need coherence – the property that, no matter the derivation, the terms resulting from the translation are equal.

▶ **Theorem 21** (Coherence). *For every two derivations D_1, D_2 of the same typing judgment $\Gamma\vdash e:\tau\ \sigma$ $\lambda^{\leq}_{\$}$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e]\!]_{D_1}=\mathcal{G}[\![e]\!]_{D_2}$.*

Proof. In the appendix. ◀

We can now prove soundness for $\lambda^{\leq}_{\$}$ with respect to the typed CGS translation:

▶ **Lemma 22.** *Suppose that for some two $\lambda^{\leq}_{\$}$ terms e_1 and e_2 we have $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$. Then for every two derivations D_1 and D_2 for $\Gamma\vdash e_1:\tau\ \sigma$ and $\Gamma\vdash e_2:\tau\ \sigma$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e_1]\!]_{D_1}=\mathcal{G}[\![e_2]\!]_{D_2}$.*

As in the untyped case, we can prove that the target terms of the typed CGS translation are equal in $\lambda^{\leq}_{\$}$ to the source terms:

▶ **Lemma 23.** *For every derivation D of $\Gamma\vdash e:\tau\ \sigma$ we have $\lambda^{\rightarrow}_{\$};\Gamma\vdash e=\mathcal{G}[\![e]\!]_D$.*

Every equality in $\lambda^{\rightarrow}_{\$}$ is also valid in $\lambda^{\leq}_{\$}$:

▶ **Lemma 24.** *If $\lambda^{\rightarrow}_{\$};\Gamma\vdash e_1=e_2$ and $\Gamma\vdash e_1:\tau\ \sigma$, then $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$.*

We can now prove completeness of $\lambda^{\leq}_{\$}$ axioms with respect to $\lambda^{\rightarrow}_{\$}$:

▶ **Lemma 25.** *If D_1 and D_2 are derivations of $\Gamma\vdash e_1:\tau\ \sigma$ and $\Gamma\vdash e_2:\tau\ \sigma$, and $\lambda^{\rightarrow}_{\$};\Gamma\vdash\mathcal{G}[\![e_1]\!]_{D_1}=\mathcal{G}[\![e_2]\!]_{D_2}$, then $\lambda^{\leq}_{\$};\Gamma\vdash e_1=e_2:\tau\ \sigma$.*

$$\frac{\mathrm{lnf}\$(\Gamma, x : \tau_1)\ e : \tau_2\ \sigma}{\mathrm{lnf}\$(\Gamma)\ \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2} \qquad \frac{\mathrm{lnf}\$(\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2)\ e : \tau\ \sigma'}{\mathrm{lnf}\$(\Gamma)\ \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau\ \sigma'} \qquad \frac{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e : \alpha}{\mathrm{lnf}\$(\Gamma)\ e : \alpha}$$

$$\frac{}{\mathrm{lnf}\$^{\downarrow}(\Gamma, x : \tau)\ x : \tau} \qquad \frac{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \mathrm{lnf}\$(\Gamma)\ e_2 : \tau_1}{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1\ e_2 : \tau_2\ \sigma}$$

$$\frac{\mathrm{lnf}\$(\Gamma)\ e_1 : \tau_1 \xrightarrow{\sigma} \tau_2 \qquad \mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_2 : \tau_1\ [\tau_2\ \sigma]\ \tau\ \sigma'}{\mathrm{lnf}\$^{\downarrow}(\Gamma)\ e_1\ \$\ e_2 : \tau\ \sigma'}$$

Figure 10 $-beta eta long form.

5.4 From $\lambda^{\to}_{\$}$ to $\lambda^{\to}$

We can easily prove soundness of the typed CGS axioms with respect to $\lambda^{\to}$:

▶ **Lemma 26.** *For any two $\lambda^{\to}_{\$}$ terms e_1, e_2 and any typing environment Γ such that $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$ we have $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$.*

Proving completeness of $\lambda^{\to}_{\$}$ is done differently than in the untyped case. We still get unwanted redexes when translating with $\mathcal{P}[\![\cdot]\!]$. For example, if $\Gamma \vdash e : \tau_1\ [\tau_2\ \sigma]\ \tau'\ \sigma'$, then $\Gamma \vdash \lambda x.\, x\ \$\ e : (\tau_1 \xrightarrow{\sigma} \tau_2) \xrightarrow{\sigma'} \tau'$ – but their translations are equal in $\lambda^{\to}$:

$$\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![\lambda x.\, x\ \$\ e]\!] = \lambda x.\, \mathcal{P}[\![e]\!]\ x = \mathcal{P}[\![e]\!]$$

The problem exists only for eta reductions and beta expansions: every beta reduction in the translated term corresponds to a β_p or $\beta^p_{\$}$ reduction in the source term, and similarly, every eta expansion in the translated term corresponds to a η_p or $\eta_{\$}$ expansion in the source term.[1] We can use beta eta long forms [1] to solve this issue. Let us define $\mathrm{lnf}(\Gamma)\ e : \tau$ to mean "in the type environment Γ the expression e has type τ and is in beta eta long form". We have the following:

▶ Property 27. If $\lambda^{\to}; \Gamma \vdash e_1 = e_2$, then there exists a $\lambda^{\to}$ term e, in beta eta long form, such that e_1 and e_2 both reduce to e using only beta reductions and eta expansions.

We can easily prove the following:

▶ **Lemma 28.** *If $\lambda^{\to}_{\$}$ term e_1 is typable in Γ and $\mathcal{P}[\![e_1]\!]$ reduces to e in $\mathcal{C}[\![\Gamma]\!]$ using only beta reductions and eta expansions, then there exists a $\lambda^{\to}_{\$}$ term e_2 such that $e = \mathcal{P}[\![e_2]\!]$ and $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$.*

But this lemma alone cannot be used to prove completeness. Applying the lemma to the two reduction sequences from Property 27 give us two $\lambda^{\to}_{\$}$ terms which translate to the same $\lambda^{\to}$ term, but we do not know yet if they are equal. Fortunately, we can use the fact that their translation is in beta eta long form to give a positive answer to this question.

Let us begin with presenting the syntax of the $\lambda^{\to}_{\$}$ analogue of the beta eta long forms, we call them $-beta eta long forms (Figure 10). We prove that if the translated $\lambda^{\to}_{\$}$ term is in the beta eta long form, then the original term is in the $-beta eta long form:

▶ **Lemma 29.** *If $\mathit{lnf}(\mathcal{C}[\![\Gamma]\!])\ \mathcal{P}[\![e]\!] : \mathcal{C}[\![\tau\ \sigma]\!]$, then $\mathit{lnf}\$(\Gamma)\ e : \tau\ \sigma$.*

[1] Conventionally, we define β_p, $\beta^p_{\$}$, η_p and $\eta_{\$}$ reductions as left-to-right directed versions of the corresponding equations.

Then we prove that if we have two $\lambda^{\to}_{\$}$ terms in \$-beta eta long forms which translate to the same $\lambda^{\to}$ term, then they are (syntactically) equal:

▶ **Lemma 30.** *If* $\mathit{lnf}\$(\Gamma)\ e_1 : \tau\ \sigma$, $\mathit{lnf}\$(\Gamma)\ e_2 : \tau\ \sigma$ *and* $\mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, *then* $e_1 = e_2$.

Now we can prove completeness of $\lambda^{\to}_{\$}$:

▶ **Lemma 31.** *If we have* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!]$, *then* $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e_2$.

Proof. Suppose that $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{P}[\![e_1]\!] = \mathcal{P}[\![e_2]\!] : \tau\ \sigma$. Using Property 27 we get a $\lambda^{\to}$ term e in eta long form such that both $\mathcal{P}[\![e_1]\!]$ and $\mathcal{P}[\![e_2]\!]$ reduce to e using only beta reductions and eta expansions. Applying Lemma 28 we get $\lambda^{\to}_{\$}; \Gamma \vdash e_1 = e'_1$, $\lambda^{\to}_{\$}; \Gamma \vdash e_2 = e'_2$ and $\mathcal{P}[\![e'_1]\!] = \mathcal{P}[\![e'_2]\!] = e$. By Lemma 29 we get that e'_1 and e'_2 are in \$-beta eta long form. Lemma 30 gives us $e'_1 = e'_2$, which finishes the proof. ◀

We can now prove soundness and completeness for $\lambda^{\leq}_{\$}$:

▶ **Theorem 32** (Soundness). *Suppose that for some two* $\lambda^{\leq}_{\$}$ *terms* e_1 *and* e_2 *we have* $\lambda^{\leq}_{\$}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$. *Then for every two derivations* D_1 *and* D_2 *for* $\Gamma \vdash e_1 : \tau\ \sigma$ *and* $\Gamma \vdash e_2 : \tau\ \sigma$ *we have* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e_1]\!]_{D_1} = \mathcal{C}[\![e_2]\!]_{D_2}$.

▶ **Theorem 33** (Completeness). *For every two derivations* D_1, D_2 *of* $\Gamma \vdash e_1 \tau\ \sigma$ *and* $\Gamma \vdash e_2 \tau\ \sigma$, *if* $\lambda^{\to}; \mathcal{C}[\![\Gamma]\!] \vdash \mathcal{C}[\![e_1]\!]_{D_1} = \mathcal{C}[\![e_2]\!]_{D_2}$, *then* $\lambda^{\leq}_{\$}; \Gamma \vdash e_1 = e_2 : \tau\ \sigma$.

6 Related work

6.1 Kameyama and Hasegawa's axioms for *shift*/*reset*

We can express the *shift*/*reset* control operators in $\lambda_{\mathcal{S}_0}$ by leaving the occurrences of *reset* without changes and replacing the occurrences of the *shift* operator $\mathcal{S}k.\, e$ with $\mathcal{S}_0 k.\, \langle e \rangle$. It is an interesting question if the axioms of Kameyama and Hasegawa [6] can be validated in this embedding using the axioms for $\lambda_{\mathcal{S}_0}$.

The answer is negative. The axioms *reset-lift*, $\mathcal{S}$*-elim* and $\mathcal{S}$*-reset* cannot be validated. The reason is that the *shift*$_0$ operator, in contrast to *shift*, can reach beyond the nearest delimiter; the three equations above significantly change the structure of delimiters, which can be distinguished by repeated uses of *shift*$_0$.

Our axioms β_v, η_v, β_Ω and $\langle v \rangle$ are identical to corresponding Kameyama and Hasegawa's axioms. The axiom $\langle \mathcal{S}_0 \rangle$ is taken from the reduction semantics for *shift*$_0$. The remaining two axioms $\eta_{\langle\cdot\rangle}$ and $\langle \lambda \rangle$ are different, but are related to $\mathcal{S}$*-elim* and *reset-lift*.

The Kameyama and Hasegawa's $\mathcal{S}$*-elim* axiom $\mathcal{S}k.\, k\, e = e$ is unsound in $\lambda_{\mathcal{S}_0}$. To see why, take a look at the $\lambda_{\mathcal{S}_0}$ term $\langle f\, \langle g\, e \rangle \rangle$. If we apply the $\mathcal{S}$-elim-derived equality $\mathcal{S}_0 k.\, \langle k\, e \rangle = e$ right-to-left on g, we get:

$$\langle f\, \langle (\mathcal{S}_0 k.\, \langle k\, g \rangle)\, e \rangle \rangle \to \langle f\, \langle (\lambda x.\, \langle x\, e \rangle)\, g \rangle \rangle \to \langle f\, \langle \langle g\, e \rangle \rangle \rangle$$

We see that one of the *reset*$_0$'s got duplicated. The *reset*$_0$ operator is not idempotent, so the equation must be unsound. To fix the equation, we need to ensure the superfluous *reset*$_0$ gets removed in the course of evaluation. This way we obtain the axiom $\eta_{\langle\cdot\rangle}$. Let us check this using our previous example:

$$\begin{aligned}&\langle f\, \langle (\mathcal{S}_0 k.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, g \rangle)\, e \rangle \rangle \to \langle f\, \langle (\lambda x.\, \mathcal{S}_0 z.\, (\lambda y.\, \langle y\, e \rangle)\, x)\, g \rangle \rangle \\ &\to \langle f\, \langle \mathcal{S}_0 z.\, (\lambda y.\, \langle y\, e \rangle)\, g \rangle \rangle \to \langle f\, ((\lambda y.\, \langle y\, e \rangle)\, g) \rangle \to \langle f\, \langle g\, e \rangle \rangle\end{aligned}$$

The Kameyama and Hasegawa's *reset-lift* axiom $\langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle$ is invalid in $\lambda_{\mathcal{S}_0}$. Notice that the main fact stated by the axiom is that the subexpression e_1 is always evaluated in an empty context. The same fact is the basis for the $\langle\lambda\rangle$ axiom.

6.2 Kameyama and Hasegawa's axioms in the typed setting

It is shown in [8] that the typed *shift*/*reset* [2] can be embedded in $\lambda^{\leq}_{\mathcal{S}_0}$ so that the type-directed CPS translation for this embedding gives terms which are beta eta equal to the standard CPS translation for *shift*/*reset*. This means that the axioms of Kameyama and Hasegawa are validated for this embedding by the axioms for $\lambda^{\leq}_{\mathcal{S}_0}$.

How is this possible, even though the embedding is the same on the term level as the untyped embedding? The answer, of course, lies in the types. Consider, for example, the Kameyama and Hasegawa's *reset-lift* axiom $(\langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle)$. In the untyped setting, this is obviously invalid – the left hand side and the right hand side have obviously different structure of delimiters, which can be distinguished by two shift_0's. But in the typed setting, we know that the types in the derivations generated by the embedding are shallow: the typing annotations are only of the form ϵ or $[\tau_1]\,\tau_2$. This means that any term in the target of the embedding which has the form $\langle e\rangle$ has a pure typing. So the following is derivable (we use the β_p axiom):

$$\lambda^{\leq}_{\mathcal{S}_0};\Gamma \vdash \langle(\lambda x.\, e_1)\,\langle e_2\rangle\rangle = \langle e_1[\langle e_2\rangle/x]\rangle = (\lambda x.\,\langle e_1\rangle)\,\langle e_2\rangle$$

Let us see another example – the $\mathcal{S}$-elim axiom $(\mathcal{S}k.\, k\, e = e)$. It is embedded into $\lambda^{\leq}_{\mathcal{S}_0}$ in the form $\mathcal{S}_0 k.\,\langle k\, e\rangle = e$. We have the following:

$$\begin{aligned}\lambda^{\leq}_{\mathcal{S}_0};\Gamma \vdash \mathcal{S}_0 k.\,\langle k\, e\rangle &= \mathcal{S}_0 k.\,\langle(\lambda x.\, k\, x)\, e\rangle = \mathcal{S}_0 k.\,\langle(\lambda x.\, \mathcal{S}_0 z.\,\langle k\, x\rangle)\, e\rangle \\ &= \mathcal{S}_0 k.\,\langle(\lambda x.\, \mathcal{S}_0 z.\, k\, x)\, e\rangle = e\end{aligned}$$

We used, in sequence, the axioms η_p, $\langle\lambda\rangle$, $\langle p\rangle$ and $\eta_{\langle\cdot\rangle}$. Notice that the use of the axiom $\langle p\rangle$ was correct only because the return type of k was pure. So the equality is not valid in general, but it is valid in the image of the embedding of typed *shift*/*reset*.

The conclusion is as follows: the type system of $\lambda^{\leq}_{\mathcal{S}_0}$ tracks how the program accesses the context stack, and the typed axioms can use the type information to make some reasoning valid which is not valid in general. The typed axioms of Kameyama and Hasegawa are valid in $\lambda^{\leq}_{\mathcal{S}_0}$ when the type annotations are shallow.

6.3 Connection with the axioms for the CPS Hierarchy

We can embed the CPS Hierarchy λ_H [3] in the calculus $\lambda_\$$, as shown in [9]. A natural question is whether Kameyama's axioms for the CPS Hierarchy [5] are validated by the axioms for $\lambda_\$$. The answer is yes. Let $\mathcal{C}_H[\![\cdot]\!]$ be the CPS translation for the CPS Hierarchy and $\mathcal{H}[\![\cdot]\!]$ be the embedding of λ_H inside $\lambda_\$$. In [9] it is proven that $\lambda \vdash \mathcal{C}_H[\![e]\!] = \mathcal{C}[\![\mathcal{H}[\![e]\!]]\!]$. So the following sequence of equivalences is true:

$$\lambda_H \vdash e_1 = e_2 \Leftrightarrow \lambda \vdash \mathcal{C}_H[\![e_1]\!] = \mathcal{C}_H[\![e_2]\!] \Leftrightarrow \lambda \vdash \mathcal{C}[\![\mathcal{H}[\![e_1]\!]]\!] = \mathcal{C}[\![\mathcal{H}[\![e_2]\!]]\!] \Leftrightarrow \lambda_\$ \vdash \mathcal{H}[\![e_1]\!] = \mathcal{H}[\![e_2]\!]$$

Because Kameyama's axioms specialized for the first level coincide with the axioms of Kameyama and Hasegawa for *shift*/*reset*, the result may seem paradoxical: we said in Section 6.1 that these axioms are not valid in $\mathit{shift}_0/\mathit{reset}_0$! There is no paradox because $\mathcal{H}[\![\cdot]\!]$ gives a different embedding of *shift*/*reset* than the one used in Section 6.1:

$$\begin{aligned}\mathcal{H}[\![\mathcal{S}_1 x.\, e]\!] &= \mathcal{S}_0 k.\,\langle e[\lambda x.\, \mathcal{S}_0 f.\, \mathcal{S}_0 g.\,(\lambda y.\, g\,\$\, f\, y)\,\$\, k\, x/x]\rangle \\ \mathcal{H}[\![\langle e\rangle_1]\!] &= \mathcal{S}_0 f.\, \mathcal{S}_0 g.\,(\lambda x.\, g\,\$\, f\, x)\,\$\,\langle\mathcal{H}[\![e]\!]\rangle\end{aligned}$$

7 Conclusion

We have presented sound and complete axioms for untyped languages $\lambda_{\mathcal{S}_0}$ and $\lambda_{\$}$ and typed languages with subtyping $\lambda^{\leq}_{\mathcal{S}_0}$ and $\lambda^{\leq}_{\$}$. In future work we will explore polymorphic and call-by-name variants of the languages considered.

Acknowledgments. Many thanks to Dariusz Biernacki, Maciej Piróg and the anonymous referees for their valuable comments. This work was funded by Polish NCN grant DEC-2011/03/B/ST6/00348, and co-funded by the European Social Fund.

References

1 Henk Barendregt. Lambda calculi with types. In Samson Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 2*, chapter 2, pages 118–309. Oxford University Press, Oxford, 1992.

2 Olivier Danvy and Andrzej Filinski. A functional abstraction of typed contexts. DIKU Rapport 89/12, DIKU, Computer Science Department, University of Copenhagen, Copenhagen, Denmark, July 1989.

3 Olivier Danvy and Andrzej Filinski. Abstracting control. In Mitchell Wand, editor, *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming*, pages 151–160, Nice, France, June 1990. ACM Press.

4 Paul Downen and Zena M. Ariola. A systematic approach to delimited control with multiple prompts. In Helmut Seidl, editor, *ESOP'12*, Lecture Notes in Computer Science, pages 234–253, Tallinn, Estonia, April 2012. Springer-Verlag.

5 Yukiyoshi Kameyama. Axioms for control operators in the CPS hierarchy. *Higher-Order and Symbolic Computation*, 20(4):339–369, 2007. A preliminary version was presented at the Fourth ACM SIGPLAN Workshop on Continuations (CW'04).

6 Yukiyoshi Kameyama and Masahito Hasegawa. A sound and complete axiomatization of delimited continuations. In Olin Shivers, editor, *ICFP'03*, SIGPLAN Notices, Vol. 38, No. 9, pages 177–188, Uppsala, Sweden, August 2003. ACM Press.

7 Oleg Kiselyov and Chung-chieh Shan. A substructural type system for delimited continuations. In Simona Ronchi Della Rocca, editor, *TLCA'07*, number 4583 in Lecture Notes in Computer Science, pages 223–239, Paris, France, June 2007. Springer-Verlag.

8 Marek Materzok and Dariusz Biernacki. Subtyping delimited continuations. In Oliver Danvy, editor, *ICFP'11*, pages 81–93, Tokyo, Japan, September 2011. ACM Press.

9 Marek Materzok and Dariusz Biernacki. A dynamic interpretation of the CPS hierarchy. In Ranjit Jhala and Atsushi Igarashi, editors, *APLAS'12*, number 7705 in Lecture Notes in Computer Science, pages 296–311, Kyoto, Japan, December 2012.

10 Amr Sabry. *The Formal Relationship between Direct and Continuation-Passing Style Optimizing Compilers: A Synthesis of Two Paradigms*. PhD thesis, Computer Science Department, Rice University, Houston, Texas, August 1994. Technical report 94-242.

11 Amr Sabry. Note on axiomatizing the semantics of control operators. Technical Report CIS-TR-96-03, Department of Computer and Information Science, University of Oregon, 1996.

12 Amr Sabry and Matthias Felleisen. Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360, 1993. A preliminary version was presented at the 1992 ACM Conference on Lisp and Functional Programming (LFP 1992).

A Useful lemmas

▶ **Lemma 34** ($\$_{R\beta}$). $\lambda_{\$} \vdash k \,\$\, (\lambda x.\, e_1) \, e_2 = (\lambda x.\, k \,\$\, e_1) \,\$\, e_2$

Proof. $k \,\$\, (\lambda x.\, e_1) \, e_2 \overset{\$_E}{=} (\lambda x.\, k \,\$\, (\lambda x.\, e_1) \, x) \,\$\, e_2 \overset{\beta_v}{=} (\lambda x.\, k \,\$\, e_1) \,\$\, e_2$ ◀

▶ **Definition 35** (CGS translation of values).

$$\mathcal{G}_v[\![x]\!] \;=\; x \qquad \mathcal{G}_v[\![\lambda x.\, e]\!] \;=\; \lambda x.\, \mathcal{G}[\![e]\!]$$

▶ **Lemma 36.** $\mathcal{G}[\![v]\!] = \mathcal{S}_0 k.\, k \, \mathcal{G}_v[\![v]\!]$

▶ **Lemma 37.** *$\lambda_{\$}$ axiom $\$_E$ is equivalent to the following three equations:*

$$\begin{aligned} v \,\$\, e_1 \, e_2 &= (\lambda x.\, v \,\$\, x \, e_2) \,\$\, e_1 && (\$_L) \\ v \,\$\, v' \, e &= (\lambda x.\, v \,\$\, v' \, x) \,\$\, e && (\$_R) \\ v \,\$\, e_1 \,\$\, e_2 &= (\lambda x.\, v \,\$\, x \,\$\, e_2) \,\$\, e_1 && (\$_{\$}) \end{aligned}$$

Proof. Equations $\$_L$, $\$_R$ and $\$_{\$}$ are obviously instances of $\$_E$. In the other direction the proof is by induction on the context E.

- $E = \bullet$

$$v \,\$\, e \overset{\eta_v}{=} (\lambda x.\, v \, x) \,\$\, e \overset{\$_v}{=} (\lambda x.\, v \,\$\, x) \,\$\, e$$

- $E = E' \, e$

$$\begin{aligned} & v \,\$\, (E' \, e)[e'] \overset{\text{def}}{=} v \,\$\, E'[e' \, e] \overset{\text{ind}}{=} (\lambda x.\, v \,\$\, E'[x]) \,\$\, e' \, e \overset{\$_L}{=} (\lambda x.\, (\lambda x.\, x \,\$\, E'[x]) \,\$\, x \, e) \,\$\, e' \\ \overset{\text{ind}}{=} \;& (\lambda x.\, v \,\$\, E'[x \, e]) \,\$\, e' \overset{\text{def}}{=} (\lambda x.\, v \,\$\, (E' \, e)[x]) \,\$\, e' \end{aligned}$$

The other two cases are similar. ◀

B Proof of Property 1

1-4 proven by induction on the expression e. Only the nontrivial cases are shown.

1. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] = e$.

$$\begin{aligned} & \mathcal{D}^{-1}[\![\mathcal{D}[\![\langle e \rangle]\!]]\!] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![(\lambda x.\, x) \,\$\, \mathcal{D}[\![e]\!]]\!] \overset{\text{def}}{=} (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}^{-1}[\![\mathcal{D}[\![e]\!]]\!] \rangle) \, (\lambda x.\, x) \\ \overset{\text{ind}}{=} \;& (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e \rangle) \, (\lambda x.\, x) \overset{\beta_v}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, x) \, e \rangle \overset{\langle v \rangle}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \langle x \rangle) \, e \rangle \\ \overset{\langle \lambda \rangle}{=} \;& \langle (\lambda x.\, x) \, e \rangle \overset{\beta_\Omega}{=} \langle e \rangle \end{aligned}$$

2. For every $\lambda_{\$}$ term e we have $\lambda_{\$} \vdash \mathcal{D}[\![\mathcal{D}^{-1}[\![e]\!]]\!] = e$.

$$\begin{aligned} & \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1 \,\$\, e_2]\!]]\!] \overset{\text{def}}{=} \mathcal{D}[\![(\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}^{-1}[\![e_2]\!] \rangle) \, \mathcal{D}^{-1}[\![e_1]\!]]\!] \\ \overset{\text{def}}{=} \;& (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, \mathcal{D}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]) \, \mathcal{D}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!] \\ \overset{\text{ind}}{=} \;& (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \, e_1 \\ \overset{\eta_{\$}}{=} \;& \mathcal{S}_0 k.\, k \,\$\, (\lambda f.\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \, e_1 \\ \overset{\$_{R\beta}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, x) \,\$\, (\lambda x.\, \mathcal{S}_0 z.\, f \, x) \, e_2) \,\$\, e_1 \\ \overset{\$_{R\beta}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, (\lambda x.\, x) \,\$\, \mathcal{S}_0 z.\, f \, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\beta_{\$}}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, (\lambda x.\, f \, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\eta_v}{=} \;& \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, e_2) \,\$\, e_1 \overset{\$_E}{=} \mathcal{S}_0 k.\, k \,\$\, e_1 \,\$\, e_2 \overset{\eta_{\$}}{=} e_1 \,\$\, e_2 \end{aligned}$$

3. For every $\lambda_{\mathcal{S}_0}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]$.

$$\begin{aligned}
& \mathcal{C}[\![\langle e\rangle]\!] \overset{\text{def}}{=} \mathcal{C}[\![e]\!]\,(\lambda x.\,\lambda k.\,k\,x) \overset{\eta}{=} \lambda k.\,\mathcal{C}[\![e]\!]\,(\lambda x.\,\lambda k.\,k\,x)\,k \\
\overset{\beta}{=}\ & \lambda k.\,(\lambda f.\,\mathcal{C}[\![e]\!]\,f\,k)\,(\lambda x.\,\lambda k.\,k\,x) \overset{\beta}{=} \lambda k.\,(\lambda k'.\,k'\,(\lambda x.\,\lambda k.\,k\,x))\,(\lambda f.\,\mathcal{C}[\![e]\!]\,f\,k) \\
\overset{\text{ind}}{=}\ & \lambda k.\,(\lambda k'.\,k'\,(\lambda x.\,\lambda k.\,k\,x))\,(\lambda f.\,\mathcal{C}[\![\mathcal{D}[\![e]\!]]\!]\,f\,k) \\
\overset{\text{def}}{=}\ & \mathcal{C}[\![(\lambda x.\,x)\,\$\,\mathcal{D}[\![e]\!]]\!] \overset{\text{def}}{=} \mathcal{C}[\![\mathcal{D}[\![\langle e\rangle]\!]]\!]
\end{aligned}$$

4. For every $\lambda_{\$}$ term e we have $\lambda \vdash \mathcal{C}[\![e]\!] = \mathcal{C}[\![\mathcal{D}^{-1}[\![e]\!]]\!]$.

$$\begin{aligned}
& \mathcal{C}[\![e_1\,\$\,e_2]\!] \overset{\text{def}}{=} \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,f\,k) \\
\overset{\eta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,\lambda k.\,f\,x\,k)\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,(\lambda z.\,\lambda k.\,f\,x\,k)\,(\lambda x.\,\lambda k.\,k\,x))\,k) \\
\overset{\text{def}}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!]\,(\lambda x.\,\lambda k.\,k\,x))\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![e_1]\!]\,(\lambda f.\,(\lambda k'.\,\mathcal{C}[\![e_2]\!]\,(\lambda x.\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!])\,x\,k'))(\lambda x.\,\lambda k.\,k\,x)\,k) \\
\overset{\text{ind}}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda f.\,(\lambda k'.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_2]\!]]\!]\,(\lambda x.\,(\lambda x.\,\mathcal{C}[\![\mathcal{S}_0 z.\,f\,x]\!])\,x\,k'))(\lambda x.\,\lambda k.\,k\,x)\,k) \\
\overset{\text{def}}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda f.\,\mathcal{C}[\![\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle]\!]\,k) \\
\overset{\beta}{=}\ & \lambda k.\,\mathcal{C}[\![\mathcal{D}^{-1}[\![e_1]\!]]\!]\,(\lambda x.\,(\lambda f.\,\mathcal{C}[\![\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle]\!])\,x\,k) \\
\overset{\text{def}}{=}\ & \mathcal{C}[\![(\lambda f.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,f\,x)\,\mathcal{D}^{-1}[\![e_2]\!]\rangle)\,\mathcal{D}^{-1}[\![e_1]\!]]\!] \overset{\text{def}}{=} \mathcal{C}[\![\mathcal{D}^{-1}[\![e_1\,\$\,e_2]\!]]\!]
\end{aligned}$$

5. $\lambda_{\mathcal{S}_0} \vdash e_1 = e_2$ implies $\lambda_{\$} \vdash \mathcal{D}[\![e_1]\!] = \mathcal{D}[\![e_2]\!]$.

- (β_Ω) $(\lambda x.\,E[x])\,e = E[e]$

$$\begin{aligned}
& \mathcal{D}[\![(\lambda x.\,E[x])\,e]\!] \overset{\text{def}}{=} (\lambda x.\,(\mathcal{D}[\![E]\!])[x])\,\mathcal{D}[\![e]\!] \\
\overset{\eta_\$}{=}\ & \mathcal{S}_0 k.\,k\,\$\,(\lambda x.\,(\mathcal{D}[\![E]\!])[x])\,\mathcal{D}[\![e]\!] \\
\overset{\$_{R\beta}}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,k\,\$\,(\mathcal{D}[\![E]\!])[x])\,\$\,\mathcal{D}[\![e]\!] \overset{\$_E}{=} \mathcal{S}_0 k.\,k\,\$\,(\mathcal{D}[\![E]\!])[\mathcal{D}[\![x]\!]] \\
\overset{\text{def}}{=}\ & \mathcal{S}_0 k.\,k\,\$\,\mathcal{D}[\![E[x]]\!] \overset{\eta_\$}{=} \mathcal{D}[\![E[x]]\!]
\end{aligned}$$

- $(\langle\mathcal{S}_0\rangle)$ $\langle E[\mathcal{S}_0 x.\,e]\rangle = e[\lambda x.\,\langle E[x]\rangle/x]$

$$\begin{aligned}
& \mathcal{D}[\![\langle E[\mathcal{S}_0 x.\,e]\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[\mathcal{S}_0 x.\,\mathcal{D}[\![e]\!]] \\
\overset{\$_E}{=}\ & (\lambda x.\,(\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[x])\,\$\,\mathcal{S}_0 x.\,\mathcal{D}[\![e]\!] \\
\overset{\beta_\$}{=}\ & \mathcal{D}[\![e]\!][\lambda x.\,(\lambda x.\,x)\,\$\,(\mathcal{D}[\![E]\!])[x]/x] \overset{\text{def}}{=} \mathcal{D}[\![e[\lambda x.\,\langle E[x]\rangle/x]]\!]
\end{aligned}$$

- $(\langle v\rangle)$ $\langle v\rangle = v$

$$\mathcal{D}[\![\langle v\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,\mathcal{D}[\![v]\!] \overset{\$_v}{=} (\lambda x.\,x)\,\mathcal{D}[\![v]\!] \overset{\beta_v}{=} \mathcal{D}[\![v]\!]$$

- $(\eta_{\langle\cdot\rangle})$ $\mathcal{S}_0 k.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,e\rangle = e$

$$\begin{aligned}
& \mathcal{D}[\![\mathcal{S}_0 k.\,\langle(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,e\rangle]\!] \overset{\text{def}}{=} \mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{S}_0 z.\,k\,x)\,\mathcal{D}[\![e]\!] \\
\overset{\$_{R\beta}}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{S}_0 z.\,k\,x)\,\$\,\mathcal{D}[\![e]\!] \\
\overset{\beta_\$}{=}\ & \mathcal{S}_0 k.\,(\lambda x.\,k\,x)\,\$\,\mathcal{D}[\![e]\!] \overset{\eta_v}{=} \mathcal{S}_0 k.\,k\,\$\,\mathcal{D}[\![e]\!] \overset{\eta_\$}{=} \mathcal{D}[\![e]\!]
\end{aligned}$$

- $(\langle\lambda\rangle)$ $\langle(\lambda x.\,\mathcal{S}_0 k.\,\langle e_1\rangle)\,e_2\rangle = \langle(\lambda x.\,e_1)\,e_2\rangle$

$$\begin{aligned}
& \mathcal{D}[\![\langle(\lambda x.\,\mathcal{S}_0 k.\,\langle e_1\rangle)\,e_2\rangle]\!] \overset{\text{def}}{=} (\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\mathcal{D}[\![e_2]\!] \\
\overset{\$_{R\beta}}{=}\ & (\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{S}_0 k.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\$\,\mathcal{D}[\![e_2]\!] \overset{\beta_\$}{=} (\lambda x.\,(\lambda x.\,x)\,\$\,\mathcal{D}[\![e_1]\!])\,\$\,\mathcal{D}[\![e_2]\!] \\
\overset{\$_{R\beta}}{=}\ & (\lambda x.\,x)\,\$\,(\lambda x.\,\mathcal{D}[\![e_1]\!])\,\mathcal{D}[\![e_2]\!] \overset{\text{def}}{=} \mathcal{D}[\![\langle(\lambda x.\,e_1)\,e_2\rangle]\!]
\end{aligned}$$

6. $\lambda_{\$} \vdash e_1 = e_2$ implies $\lambda_{\mathcal{S}_0} \vdash \mathcal{D}^{-1}[\![e_1]\!] = \mathcal{D}^{-1}[\![e_2]\!]$.

- $(\beta_{\$})$ $v \,\$\, \mathcal{S}_0 x.\, e = e[v/x]$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v \,\$\, \mathcal{S}_0 x.\, e]\!] \overset{\text{def}}{=} (\lambda f.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, f\, x)\, (\mathcal{S}_0 x.\, \mathcal{D}^{-1}[\![e]\!]) \rangle)\, \mathcal{D}^{-1}[\![v]\!] \\ \overset{\beta_v}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, (\mathcal{S}_0 x.\, \mathcal{D}^{-1}[\![e]\!]) \rangle \\ \overset{\langle \mathcal{S}_0 \rangle}{=} & \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, y \rangle / x] \\ \overset{\beta_v}{=} & \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \langle \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, y \rangle / x] \overset{\langle \mathcal{S}_0 \rangle}{=} \mathcal{D}^{-1}[\![e]\!][\lambda y.\, \mathcal{D}^{-1}[\![v]\!]\, y / x] \\ \overset{\eta_v}{=} & \mathcal{D}^{-1}[\![e]\!][\mathcal{D}^{-1}[\![v]\!]/x] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![e[v/x]]\!] \end{array}$$

- $(\eta_{\$})$ $\mathcal{S}_0 x.\, x \,\$\, e = e$

$$\mathcal{D}^{-1}[\![\mathcal{S}_0 x.\, x \,\$\, e]\!] \overset{\text{def}}{=} \mathcal{S}_0 x.\, \langle (\lambda y.\, \mathcal{S}_0 z.\, x\, y)\, \mathcal{D}^{-1}[\![e]\!] \rangle \overset{\eta_{\langle\cdot\rangle}}{=} \mathcal{D}^{-1}[\![e]\!]$$

- $(\$_v)$ $v_1 \,\$\, v_2 = v_1\, v_2$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v_1 \,\$\, v_2]\!] \overset{\text{def}}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v_1]\!]\, x)\, \mathcal{D}^{-1}[\![v_2]\!] \rangle \overset{\beta_v}{=} \langle \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v_1]\!]\, \mathcal{D}^{-1}[\![v_2]\!] \rangle \\ \overset{\langle \mathcal{S}_0 \rangle}{=} & \mathcal{D}^{-1}[\![v_1]\!]\, \mathcal{D}^{-1}[\![v_2]\!] \overset{\text{def}}{=} \mathcal{D}^{-1}[\![v_1\, v_2]\!] \end{array}$$

- $(\$_E)$ $v \,\$\, E[e] = (\lambda x.\, v \,\$\, E[x]) \,\$\, e$

$$\begin{array}{rl} & \mathcal{D}^{-1}[\![v \,\$\, E[e]]\!] \overset{\text{def}}{=} \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, ((\mathcal{D}^{-1}[\![E]\!])[\mathcal{D}^{-1}[\![e]\!]]) \rangle \\ \overset{\beta_\Omega}{=} & \langle (\lambda x.\, (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, ((\mathcal{D}^{-1}[\![E]\!])[x]))\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\text{def}}{=} & \langle (\lambda x.\, (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!])\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\langle \lambda \rangle}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!] \rangle)\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\beta_v}{=} & \langle (\lambda x.\, \mathcal{S}_0 z.\, (\lambda x.\, \langle (\lambda x.\, \mathcal{S}_0 z.\, \mathcal{D}^{-1}[\![v]\!]\, x)\, \mathcal{D}^{-1}[\![E[x]]\!] \rangle)\, x)\, \mathcal{D}^{-1}[\![e]\!] \rangle \\ \overset{\text{def}}{=} & \mathcal{D}^{-1}[\![(\lambda x.\, v \,\$\, E[x]) \,\$\, e]\!] \end{array}$$

C Proof of Lemma 6

For every $\lambda_{\$}$ term e we have $\lambda_{\$} \vdash e = \mathcal{G}[\![e]\!]$.

Proof is by induction on the expression e. Only the nontrivial cases are presented.

- $e = \lambda x.\, e'$

$$\lambda x.\, e' \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, \lambda x.\, e' \overset{\$_v}{=} \mathcal{S}_0 k.\, k\, (\lambda x.\, e') \overset{\text{ind}}{=} \mathcal{S}_0 k.\, k\, (\lambda x.\, \mathcal{G}[\![e']\!]) \overset{\text{def}}{=} \mathcal{G}[\![\lambda x.\, e']\!]$$

- $e = e_1\, e_2$

$$\begin{array}{rl} & e_1\, e_2 \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, e_1\, e_2 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f\, e_2) \,\$\, e_1 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, e_2) \,\$\, e_1 \\ \overset{\text{ind}}{=} & \mathcal{S}_0 k.\, (\lambda f.\, (\lambda x.\, k \,\$\, f\, x) \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] \overset{\text{def}}{=} \mathcal{G}[\![e_1\, e_2]\!] \end{array}$$

- $e = e_1 \,\$\, e_2$

$$\begin{array}{rl} & e_1 \,\$\, e_2 \overset{\eta_{\$}}{=} \mathcal{S}_0 k.\, k \,\$\, e_1 \,\$\, e_2 \overset{\$_E}{=} \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, e_2) \,\$\, e_1 \\ \overset{\text{ind}}{=} & \mathcal{S}_0 k.\, (\lambda f.\, k \,\$\, f \,\$\, \mathcal{G}[\![e_2]\!]) \,\$\, \mathcal{G}[\![e_1]\!] \overset{\text{def}}{=} \mathcal{G}[\![e_1 \,\$\, e_2]\!] \end{array}$$

$$\frac{}{\epsilon \ll^{\leq}} \qquad \frac{}{[\tau\ \sigma]\ \tau\ \sigma \ll^{\leq}} \qquad \frac{\sigma \ll^{\leq} \vec{\sigma}}{\sigma \ll^{\leq} \epsilon\ \vec{\sigma}} \qquad \frac{[\tau_3\ \sigma_3]\ \tau_2\ \sigma_2 \ll^{\leq} \vec{\sigma} \qquad \tau_2'\ \sigma_2' \leq \tau_2'\ \sigma_2}{[\tau_3\ \sigma_3]\ \tau_1\ \sigma_1 \ll^{\leq} [\tau_2'\ \sigma_2']\ \tau_1\ \sigma_1\ \vec{\sigma}}$$

$$\frac{}{\Gamma, x : \tau \rhd x : \tau}\ \text{VAR} \qquad \frac{\Gamma, x : \tau_1 \rhd e : \tau_2\ \sigma}{\Gamma \rhd \lambda x : \tau_1.\, e : \tau_1 \xrightarrow{\sigma} \tau_2}\ \text{ABS} \qquad \frac{\Gamma, x : \tau_1 \xrightarrow{\sigma} \tau_2 \rhd e : \tau_3\ \sigma'}{\Gamma \rhd \mathcal{S}_0 x : \tau_1 \xrightarrow{\sigma} \tau_2.\, e : \tau_1\ [\tau_2\ \sigma]\ \tau_3\ \sigma'}\ \text{SFT}$$

$$\frac{\Gamma \rhd e_1 : \tau_1 \xrightarrow{\sigma_3} \tau_2\ \sigma_1 \qquad \Gamma \rhd e_2 : \tau_1\ \sigma_2 \qquad \sigma \ll^{\leq} \sigma_1\ \sigma_2\ \sigma_3}{\Gamma \rhd e_1\ e_2 : \tau_2\ \sigma}\ \text{APP}$$

Rule for $\lambda^{\leq}_{\mathcal{S}_0}$:

$$\frac{\Gamma \rhd e : \tau''\ [\tau'\ \sigma']\ \tau\ \sigma \qquad \tau'' \leq \tau'\ \sigma'}{\Gamma \rhd \langle e \rangle : \tau\ \sigma}\ \text{RST}$$

Rule for $\lambda^{\leq}_{\$}$:

$$\frac{\Gamma \rhd e_1 : \tau_1 \xrightarrow{\sigma'} \tau_2\ \sigma_1 \qquad \Gamma \rhd e_2 : \tau_1\ [\tau_2\ \sigma']\ \tau_3\ \sigma_2 \qquad \sigma \ll^{\leq} \sigma_1\ \sigma_2}{\Gamma \rhd e_1\ \$\ e_2 : \tau_3\ \sigma}\ \text{DOL}$$

Figure 11 The type system giving minimal types for $\lambda_{\$}$ and $\lambda^{\leq}_{\$}$.

D Proof of Theorem 21 (coherence)

The language $\lambda^{\leq}_{\$}$ has minimal types in the following sense:

▶ **Lemma 38** (Minimal types). *For every e, Γ, τ, σ such that $\Gamma \vdash e : \tau\ \sigma$ there exist τ', σ' such that $\Gamma \vdash e : \tau'\ \sigma'$ and for every τ'', σ'' such that $\Gamma \vdash e : \tau''\ \sigma''$ we have $\tau'\ \sigma' \leq \tau''\ \sigma''$.*

Proof. The type system in Figure 11 gives the minimal type. This can be proven by induction on the derivation of $\Gamma \vdash e : \tau\ \sigma$. ◀

For every derivation D_M of the minimal type judgement of Figure 11 we can find a type derivation for the same type which corresponds to the derivation D_M; let us call it $\mathcal{D}(D_M)$. We can prove the following lemma:

▶ **Lemma 39.** *For every derivation D_M of $\Gamma \rhd e : \tau\ \sigma$ and every derivation D of $\Gamma \vdash e : \tau'\ \sigma'$ we have $\lambda^{\rightarrow}_{\$}; \Gamma \vdash \mathcal{G}[\![e]\!]_D = \mathcal{G}[\![\tau\ \sigma \leq \tau'\ \sigma']\!][\mathcal{G}[\![e]\!]_{\mathcal{D}(D_M)}]$.*

Proof. Induction on the derivation D. ◀

Coherence follows immediately.

E Kameyama and Hasegawa's axiomatization of shift/reset

The axioms were presented in [6]. Syntax was adapted to the one used in this paper.

$$\begin{array}{rcll}
(\lambda x.\, e)\, v & = & e[v/x] & & (\beta_v) \\
\lambda x.\, v\, x & = & v & x \notin \mathrm{V}(v) & (\eta_v) \\
(\lambda x.\, E[x])\, e & = & E[e] & x \notin \mathrm{V}(E) & (\beta_\Omega) \\
\langle v \rangle & = & v & & (\textit{reset-value}) \\
\langle (\lambda x.\, e_1)\, \langle e_2 \rangle \rangle & = & (\lambda x.\, \langle e_1 \rangle)\, \langle e_2 \rangle & & (\textit{reset-lift}) \\
\mathcal{S}k.\, k\, e & = & e & k \notin \mathrm{V}(e) & (\mathcal{S}\textit{-elim}) \\
\langle E[\mathcal{S}k.\, e] \rangle & = & e[\lambda x.\, \langle E[x] \rangle / k] & x \notin \mathrm{V}(E) & (\textit{reset-}\mathcal{S}) \\
\mathcal{S}k.\, \langle e \rangle & = & \mathcal{S}k.\, e & & (\mathcal{S}\textit{-reset})
\end{array}$$