Electronic Notes in Theoretical Informatics and Computer Science
**ENTICS**
https://entics.episciences.org
Volume 1
Proceedings of MFPS 2022

# Call-By-Name Is Just Call-By-Value with Delimited Control

Mateusz Pyzik[2,3,4]

*Institute of Computer Science*
*University of Wrocław*
*Wrocław, Poland*

**Abstract**

Delimited control operator `shift0` exhibits versatile capabilities: it can express layered monadic effects, or equivalently, algebraic effects. Little did we know it can express lambda calculus too! We present $\Lambda_\$$, a call-by-value lambda calculus extended with `shift0` and control delimiter `$` with carefully crafted reduction theory, such that the lambda calculus with beta and eta reductions can be isomorphically embedded into $\Lambda_\$$ via a right inverse of a continuation-passing style translation. While call-by-name reductions of lambda calculus can trivially simulate its call-by-value version, we show that addition of `shift0` and `$` is the golden mean of expressive power that suffices to simulate beta and eta reductions while still admitting a simulation back. As a corollary, calculi $\Lambda\mu_v$, $\lambda_\$$, $\Lambda_\$$ and $\lambda$ all correspond equationally.

*Keywords:* delimited control, continuation-passing style, reflection, lambda calculus

## 1 Introduction

Delimited control [5,3] is a computational effect capable of expressing any monadic effect in direct style [6,7]. Moreover, delimited continuations are closely related to algebraic effects [15,16], as it has been shown that delimited control, in the form of `shift0` and `$`, are mutually macro expressible with algebraic effects with deep handlers [8,14]. It is as strong as an infinite hierarchy of `shifts` [13]. Finally, `shift0` can recursively express `control0` in direct style [19].

It is a basic observation that operational semantics of call-by-value lambda calculus can be simulated by call-by-name lambda calculus. Every $\beta_v$ ($\eta_v$) reduction step can be simulated by a $\beta$ ($\eta$) step on the same term. The latter simulation is mediated by identity relation on terms. In this paper we show that there exists a relation (i.e. CPS translation $* : \Lambda_\$ \to \lambda$) between call-by-value lambda terms extended with delimited control operators and vanilla, call-by-name lambda terms that admits a simulation in both ways, hence a bisimulation. We do not prove the bisimulation directly. Instead, we show that a stronger relationship between these calculi exists, namely a reflection.

[1] Special thanks to my PhD programme supervisor Dariusz Biernacki and to my colleague and coauthor of previous works Filip Sieczkowski for invaluable support throughout the whole endeavour.
[2] Email: matp@cs.uni.wroc.pl
[3] ORCID: 0000-0002-9978-9536
[4] IPA: [matɛuʂ piʑik]

In our previous, related works we followed the programme of Sabry and Wadler [18]: to seek a Galois connection (or even better, a reflection or isomophism) between reduction theories instead of equational correspondences whenever possible, as a Galois connection is a stronger relationship between calculi that already implies an equational correspondence. In [1] we developed such relationship between `shift` and its CPS, while in [2] we have done that for `shift0`. This paper improves upon the latter of these and upon undirected axiomatisation of [11], as we prove a reflection where the image of CPS is closed under $\beta\eta$ reduction, hence a "directed axiomatisation". However, in the particular case of CPS translation for `shift0`-style delimited control it turns out that the reduction-closed image is not a proper subcalculus but the entire $\lambda$, hence every $\lambda$-term is, in fact, in continuation-composing style. A translation back (aka. direct style translation) from $\lambda$ to $\Lambda_{\$}$ exists which allows to perfectly embed lambda calculus: image of the direct style translation is a subcalculus of $\Lambda_{\$}$ isomorphic with $\lambda$-calculus.

This paper has two goals.

(i) to bring about the definitive, fine-grained operational semantics for `shift0` that can provide, a "directed axiomatisation" of delimited control.

(ii) to show that call-by-value with delimited control in the form of `shift0` is the same thing as call-by-name purely functional programming on the level of operational semantics.

The second point cannot be stressed enough. Call-by-name reduction is exactly like call-by-value with `shift0`-style delimited control. It is not the case with abortive control nor `shift` of Danvy and Filinski [3], which can be deduced from previous axiomatisation efforts [17,9], whose CPS translations reach proper subsets of $\lambda$-terms. This puts `shift0` control operator in a unique spot and as far as we know, it has not been acknowledged yet.

### 1.1 Outline

In Section 2 a new calculus of control is presented, named $\Lambda_{\$}$, with its syntax and semantics. In Section 3 we show mutual macro-expressibility (and equational correspondence) of $\Lambda_{\$}$ and $\lambda_{\$}$, an established calculus with `shift0` and `$`, to support our claim that $\Lambda_{\$}$ is a call-by-value calculus with `shift0`-style delimited control. In Section 4 we prove theorems that justify the title of this paper: $\lambda$ can be reflected into $\Lambda_{\$}$. As a corollary, calculi $\Lambda\mu_v$ [4], $\lambda_{\$}$ [11], $\Lambda_{\$}$ and $\lambda$ all correspond equationally. We conclude and discuss related and future work in Section 5.

### 1.2 Conventions

In this paper, $\equiv$ means syntactic equality modulo renaming of bound variables, while $=$ means term equivalence up to rewrite rules in a calculus; calculus is either implied by the usage or specified directly in the subscript. Given relation $r$, $r^?$ is $(\equiv) \cup r$, i.e. reflexive closure of $r$. Rewrites may occur in arbitrary contexts, including variable-binding contexts; these contexts may capture variables of a term that is plugged into. Terms are not required to be closed; it is perflectly fine for them to be open. Some classes of contexts (i.e. bindable and pure contexts) that we define may look a lot like evaluation contexts but that is not their purpose here. We are not concerned with standard reduction nor evaluation strategy, only with general reduction on open terms that may and will often be nondeterministic.

Atomic terms include variables $x$, parenthesised terms $(M)$, unary `shift0` $\mathcal{S}_0(M)$ (we call it *thaw*) and unary `$` $\$(M)$ (we call it *freeze*). Parentheses in these unary operators are mandatory and there is no space between operator and parenthesis. Plugging term in a context $C[M]$ and postfix translations $M^*, M^\dagger, M^\#, M^\natural$ [5] are left-associative and have the highest precedence. Application $M\,N$ has the next precedence and is left-associative, followed by right-associative binary `$` (known as *plug* [10] or *dollar* [13] [6]) $M \,\$\, N$. We disambiguate $M\ \$(N)$ as $M\,(\$(N))$, and $M \,\$\, (N)$ as $M \,\$\, N$ by different spacing. Syntaxes

[5] We follow the translation naming convention of Sabry and Wadler [18].

[6] The word *plug* is somewhat justified, but we feel like it is already taken (to *plug* a term in a context), so we tend to pronouce this operator *dollar*. It's arguably a bad name but there is a precedent of its use.

let $x = M \operatorname{in} N$, $\lambda\, x\,.\, N$ and $\mathcal{S}_0\, x\,.\, N$ are right-associative, bind variable $x$ in term $N$ and have the lowest precedence.

### *1.3 Theory of preorders – Galois connections and reflections*

A calculus equipped with a reduction relation can be seen as a preordered set (*proset*, a set with a reflexive and transitive relation).

**Definition 1.1** Assume $(A, \twoheadrightarrow_A)$ and $(B, \twoheadrightarrow_B)$ are prosets. A function $f : A \to B$ is monotone iff for all $x_1, x_2 \in A$, $x_1 \twoheadrightarrow_A x_2$ implies $f(x_1) \twoheadrightarrow_B f(x_2)$.

**Definition 1.2** Monotone functions $f : A \to B$ and $g : B \to A$ form a Galois connection if, and only if for all $a \in A$ and $b \in B$, $a \twoheadrightarrow_A g(b) \iff f(a) \twoheadrightarrow_B b$.

It may be helpful to think about a Galois connection as a compiler $*$ from source language $S$ to target language $T$ coupled with a decompiler #. Laws of Galois connection guarantee that compiling and decompiling are harmonised: it does not matter whether simplification is performed on source or target, or on both ends. As an equivalence, a Galois connection allows two modes of reasoning:

- *Soundness.* Every multi-step reduction in the source is valid in the target: if $M \twoheadrightarrow_S N^{\#}$, then $M^* \twoheadrightarrow_T N$.
- *Completeness.* Every multi-step reduction in the target is valid in the source: if $M^* \twoheadrightarrow_T N$, then $M \twoheadrightarrow_S N^{\#}$.

A reader familiar with (bi)simulations may notice that if a function $f$ is monotone, then $f$ is a similarity relation. When $(f, g)$ is a Galois connection, then $f$ is a bisimilarity relation. A reader somewhat familiar with category theory may recognise that preordered set is the same thing as a *thin* category. Monotone functions are functors between those categories. Galois connection is a pair of adjoint functors. A Galois connection between symmetric preorders (better known as equivalence relations) is simply an equational correspondence.

There is an alternative characterisation of a Galois connection.

**Theorem 1.3 (Equivalent definition of Galois connection)** *Monotone functions $f : A \to B$ and $g : B \to A$ form a Galois connection if, and only if*

- $a \twoheadrightarrow_A g(f(a))$ *and*
- $f(g(b)) \twoheadrightarrow_B b$.

A reflection is a bit tighter kind of a Galois connection.

**Definition 1.4** A Galois connection $(f : A \to B, g : B \to A)$ is a reflection if, and only if for all $b \in B$, $f(g(b)) \equiv b$.

When $(f : A \to B, g : B \to A)$ is a reflection we say that $B$ can be reflected into $A$. A subcalculus $(g[B], (\twoheadrightarrow_A) \cap (g[B] \times g[B]))$ is then isomorphic to $B$.

**Definition 1.5** A reflection $(f : A \to B, g : B \to A)$ is an isomorphism if, and only if for all $a \in A$, $a \equiv g(f(a))$.

### *1.4 Background: Materzok's $\lambda_\$$*

In this introductory subsection, we present syntax, reductions, axioms (Figure 1.4) and CPS translation (Figure 2) of $\lambda_\$$, as defined by Materzok [11]. It features a variable-binding control operator `shift0` and binary operator `$` that delimits continuation captured by `shift0` in the right hand side. CPS translation to lambda calculus provides denotational semantics. The axioms were proven by Materzok to be sound and complete with respect to CPS translation. $\lambda_\$$ also has small-step operational semantics, which are sound (but not complete) with respect to axioms and CPS. These semantics allow `shift0` to capture an arbitrarily long context in a single reduction step.

$$
\begin{array}{lll}
\text{term} & e & ::= v \mid e\,e' \mid \mathcal{S}_0\,x\,.\,e \mid e\,\$\,e' \\
\text{value} & v & ::= x \mid \lambda\,x\,.\,M \\
\text{pure context} & E & ::= [\,] \mid E\,e \mid v\,E \mid E\,\$\,e
\end{array}
$$

$$
\begin{array}{llll}
(\lambda\,x\,.\,e)\,v & \to_{\beta_v} & e[v/x] & \\
v\,\$\,v' & \to_{\$_v} & v\,v' & \\
v\,\$\,E[\mathcal{S}_0\,x\,.\,e] & \to_{\$/\,\mathcal{S}_0} & e[\lambda\,y\,.\,v\,\$\,E[y]/x] & \text{if } y \text{ is fresh wrt. } v, E \\
(\lambda\,x\,.\,e)\,v & =_{\beta_v} & e[v/x] & \\
\lambda\,x\,.\,v\,x & =_{\eta_v} & v & \text{if } x \text{ is fresh wrt. } v \\
v\,\$\,v' & =_{\$_v} & v\,v' & \\
v\,\$\,E[e] & =_{\$_E} & (\lambda\,y\,.\,v\,\$\,E[y])\,\$\,e & \\
v\,\$\,\mathcal{S}_0\,x\,.\,e & =_{\beta_\$} & e[v/x] & \\
\mathcal{S}_0\,x\,.\,x\,\$\,e & =_{\eta_\$} & e & \text{if } x \text{ is fresh wrt. } e
\end{array}
$$

Fig. 1. Syntax, reductions and axioms of $\lambda_\$$ [11].

$$\mathcal{C}[\![\cdot]\!] : \lambda_\$ \to \lambda$$

$$
\begin{array}{ll}
\mathcal{C}[\![v]\!] & :\equiv \lambda\,k\,.\,k\,\mathcal{C}_v[\![v]\!] \\
\mathcal{C}[\![e\,e']\!] & :\equiv \lambda\,k\,.\,\mathcal{C}[\![e]\!]\,(\lambda\,x\,.\,\mathcal{C}[\![e']\!]\,(\lambda\,y\,.\,x\,y\,k)) \\
\mathcal{C}[\![\mathcal{S}_0\,x\,.\,e]\!] & :\equiv \lambda\,x\,.\,\mathcal{C}[\![e]\!] \\
\mathcal{C}[\![e\,\$\,e']\!] & :\equiv \lambda\,k\,.\,\mathcal{C}[\![e]\!]\,(\lambda\,x\,.\,\mathcal{C}[\![e']\!]\,x\,k) \\
\mathcal{C}_v[\![x]\!] & :\equiv x \\
\mathcal{C}_v[\![\lambda\,x\,.\,e]\!] & :\equiv \lambda\,x\,.\,\mathcal{C}[\![e]\!]
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{S}_0(e) & :\equiv (\lambda\,x\,.\,\mathcal{S}_0\,k\,.\,x\,k)\,e \\
\$(e) & :\equiv \lambda\,x\,.\,x\,\$\,e
\end{array}
$$

Fig. 2. Semantics of $\lambda_\$$, described by CPS translation [11] and macro-definitions of unary variants of control operators.

**Example 1.6** Here's an evaluation example. Let $I \equiv \lambda\,x\,.\,x$.

$$
\begin{array}{c}
I\,\$\,I\,(\mathcal{S}_0\,f\,.\,f\,(f z)) \to_{\$/\,\mathcal{S}_0} (f\,(f\,z))[\lambda\,x\,.\,I\,\$\,I\,x/f] \equiv (\lambda\,x\,.\,I\,\$\,I\,x)\,((\lambda\,x\,.\,I\,\$\,I\,x)\,z) \\
\to_{\beta_v} (\lambda\,x\,.\,I\,\$\,I\,x)\,(I\,\$\,I\,z) \to_{\beta_v} (\lambda\,x\,.\,I\,\$\,I\,x)\,(I\,\$\,z) \to_{\$_v} (\lambda\,x\,.\,I\,\$\,I\,x)\,(I\,z) \\
\to_{\beta_v} (\lambda\,x\,.\,I\,\$\,I\,x)\,z \to_{\beta_v} I\,\$\,I\,z \to_{\beta_v} I\,\$\,z \to_{\$_v} I\,z \to_{\beta_v} z
\end{array}
$$

**Example 1.7** Terms in CPS get lengthy even for very small examples: $\mathcal{C}[\![I]\!] \equiv \lambda\,k_1\,.\,k_1\,(\lambda\,x\,.\,\lambda\,k_2\,.\,k_2\,x)$.

## 2 Syntax and semantics of $\Lambda_\$$

Reductions in Materzok's $\lambda_\$$ were clearly lacking. What we wanted was a set of reductions that would also stand as a complete axiomatisation of `shift0`. Our goal was to improve upon a sound but incomplete reduction theory of $\lambda_{c\$}$ [2]. To reach completeness, we took the range of that previous CPS translation and closed this set under $\beta\eta$-reduction (in a similar vein to Sabry and Felleisen [17]). To our surprise, all quirks of the grammar disappeared and the entire $\lambda$ showed up. To double check, we also closed CPS of $\lambda_{\$}$ under reduction: entire $\lambda$ again. It was now a matter of time to find an improved calculus, CPS and DS (direct style) translation. Our choice to use unary operators may be considered arbitrary by some but this choice provided us with a beautiful duality of values and terms mediated by freeze and thaw and an actual, working reflection. Old semantics had to be modified anyway and syntax could use some refreshment too. Our method was to have only one binding construct and use only local reductions to keep things as simple as possible.

We define syntax and small-step operational semantics of $\Lambda_{\$}$ in Figure 3. Figure also includes some syntactic sugar to make some idioms (like let-expressions) easier on the eye.

Indeed, let-expressions with both $\beta$ and $\eta$ rule are macro-definable. It is easy to check that $\mathrm{let}\, x = V \,\mathrm{in}\, M \twoheadrightarrow_{\Lambda_\$} M[V/x]$, $\mathrm{let}\, x = M \,\mathrm{in}\, x \twoheadrightarrow_{\Lambda_\$} M$ and also $(\lambda\, x\,.\, N)\, M \twoheadrightarrow_{\Lambda_\$} \mathrm{let}\, x = M \,\mathrm{in}\, N$. Associativity of let-expressions holds, but it is undirected: if $y$ is fresh wrt. $N$, then

$$\mathrm{let}\, x = \mathrm{let}\, y = L \,\mathrm{in}\, M \,\mathrm{in}\, N =_{\Lambda_\$} \mathrm{let}\, y = L \,\mathrm{in}\, \mathrm{let}\, x = M \,\mathrm{in}\, N.$$

Initially we aimed for a calculus with a separate let construct whose CPS is diffrent than that of a $\beta$-redex, like in $\lambda_c$ [18], $\lambda_{c\mathcal{S}}$ [1] and $\lambda_{c\$}$ [2]. However, associativity kept derailing our theorems. We were stuck, so we got rid of the let construct and replaced its uses with a pattern $\mathcal{S}_0\, k\,.(\lambda\, x\,.\, k \,\$\, N) \,\$\, M$ that behaves just like let-expressions. Due to its ubiquity in the proofs, we needed a shorthand notation; a let-expression syntax was an obvious choice.

One can think of unary `$` as reifying or freezing a computation as a value. Such a value can be then reflected or thawed by unary `shift0` (if given a nonvalue, it waits until its argument becomes a value). These operations are invertible using $\$\,\mathcal{S}_0$ and $\mathcal{S}_0\,\$$ reductions. Reductions $\$_v$ and **pure** tell us that a value $V$ freezes to $\lambda\, x\,.\, x\, V$ and can be recovered from it by thawing. The job of **bind** rule is to gives names to subcomputations and make sure contexts are managed properly. Notice that due to syntactic restriction that requires **bind**-redex to be of the form $J[P]$, there is at most one legal way to **bind**-contract:

- $\mathcal{S}_0(P)$ **bind** $\mathrm{let}\, x = P \,\mathrm{in}\, \mathcal{S}_0(x)$ is a legal **bind**-contraction,
- $\mathcal{S}_0(V)$ **bind** $\mathrm{let}\, x = V \,\mathrm{in}\, \mathcal{S}_0(x)$, $V\, W$ **bind** $\mathrm{let}\, x = W \,\mathrm{in}\, V\, x$ and $V\, W$ **bind** $\mathrm{let}\, x = V \,\mathrm{in}\, x\, W$ are illegal,
- $V\, P$ **bind** $\mathrm{let}\, x = P \,\mathrm{in}\, V\, x$ is legal while $V\, P$ **bind** $\mathrm{let}\, x = V \,\mathrm{in}\, x\, P$ is not,
- $P\, M$ **bind** $\mathrm{let}\, x = P \,\mathrm{in}\, x\, M$ is legal while $P\, Q$ **bind** $\mathrm{let}\, x = Q \,\mathrm{in}\, P\, x$ is not,

To enable further developments, Figure 2 introduces continuation-passing style translation $* : \Lambda_\$ \to \lambda$, which provides alternative, denotational semantics of $\Lambda_\$$. However, one does not need to worry about a mismatch of semantics – theorems of Section 4 are more than enough to ensure that $*$ is a sound and complete translation ($M =_{\Lambda_\$} N$ iff $M^* =_\lambda N^*$). $\lambda$-calculus that we target has both $\beta$ and $\eta$ reductions (Figure 6).

To lift the spirit even higher, confluence of $\Lambda_\$$ is proven below using definitions (translation $\# : \lambda \to \Lambda_\$$) and theorems of Section 4 but the proof is straightforward enough that it may be understandable right here. It gives us a taste of usefullness of reflections and Galois connections in general: one can transfer confluence from one calculus to another. Confluence is provided here as a sanity check of a newly introduced calculus; to avoid circular reasoning, we do not use it in proofs.

**Theorem 2.1 (Confluence)** *Reduction in $\Lambda_\$$ is confluent.*

**Proof.** For concise presentation, we chain relations and flip arguments: $M \twoheadleftarrow N$ iff $N \twoheadrightarrow M$.

(i) Assume $M_1 \twoheadleftarrow M \twoheadrightarrow M_2$.

(ii) Apply monotonicity of translation $*$: $M_1^* \twoheadleftarrow M^* \twoheadrightarrow M_2^*$.

(iii) Apply confluence in $\lambda$ (Church-Rosser theorem): there is $N \in \lambda$ such that $M_1^* \twoheadrightarrow N \twoheadleftarrow M_2^*$.

(iv) Apply monotonicity of translation $\#$: $M_1^{*\#} \twoheadrightarrow N^{\#} \twoheadleftarrow M_2^{*\#}$.

| | | | |
|---|---|---|---|
| term | $L, M, N$ | $::=$ | $V \mid P$ |
| value | $V, W$ | $::=$ | $x \mid \lambda x \,.\, M \mid \$(M)$ |
| nonvalue | $P, Q$ | $::=$ | $M\,N \mid \mathcal{S}_0(M)$ |
| bindable context | $J$ | $::=$ | $[\,]\,M \mid V\,[\,] \mid \mathcal{S}_0([\,])$ |
| pure context | $K$ | $::=$ | $[\,] \mid J[K]$ |
| reduction context | $C$ | $::=$ | $[\,] \mid \lambda x \,.\, C \mid C\,M \mid M\,C \mid \mathcal{S}_0(C) \mid \$(C)$ |

$$\mathcal{S}_0\, x \,.\, M \;:\equiv\; \mathcal{S}_0(\lambda x \,.\, M)$$
$$M \,\$\, N \;:\equiv\; \$(N)\, M$$
$$\text{let}\, x = M \,\text{in}\, N \;:\equiv\; \mathcal{S}_0\, k \,.(\lambda x \,.\, k \,\$\, N) \,\$\, M$$
$$\mapsto \;:\equiv \beta_v \cup \eta_v \cup \$_v \cup \$\,\mathcal{S}_0 \cup \mathcal{S}_0\,\$ \cup \mathbf{pure} \cup \mathbf{bind}$$

| | | | |
|---|---|---|---|
| $(\lambda x \,.\, M)\, V$ | $\beta_v$ | $M[V/x]$ | |
| $\lambda x \,.\, V\, x$ | $\eta_v$ | $V$ | if $x$ is fresh wrt. $V$ |
| $\$(V)$ | $\$_v$ | $\lambda x \,.\, x\, V$ | if $x$ is fresh wrt. $V$ |
| $\$(\mathcal{S}_0(V))$ | $\$\,\mathcal{S}_0$ | $V$ | |
| $\mathcal{S}_0(\$(M))$ | $\mathcal{S}_0\,\$$ | $M$ | |
| $\mathcal{S}_0\, x \,.\, x\, V$ | **pure** | $V$ | if $x$ is fresh wrt. $V$ |
| $J[P]$ | **bind** | let $x = P$ in $J[x]$ | if $x$ is fresh wrt. $J$ |
| $C[M]$ | $\to$ | $C[N]$ | if $M \mapsto N$ |
| $M$ | $\twoheadrightarrow$ | $N$ | if $M \to^n N, n \geq 0$ |
| $M$ | $=$ | $N$ | if $M (\to \cup \leftarrow)^n N, n \geq 0$ |

Fig. 3. Direct style calculus $\Lambda_\$$.

$$* : \Lambda_\$ \to \lambda$$

| | | | |
|---|---|---|---|
| $V^*$ | $:\equiv \lambda k \,.\, k\, V^\dagger$ | $x^\dagger$ | $:\equiv x$ |
| $J[P]^*$ | $:\equiv \lambda k \,.\, P^*\, (\lambda x \,.\, J[x]^*\, k)$ | $(\lambda x \,.\, M)^\dagger$ | $:\equiv \lambda x \,.\, M^*$ |
| $(V\, W)^*$ | $:\equiv V^\dagger\, W^\dagger$ | $\$(M)^\dagger$ | $:\equiv M^*$ |
| $\mathcal{S}_0(V)^*$ | $:\equiv V^\dagger$ | | |

Fig. 4. Translation from $\Lambda_\$$ to continuation-passing style.

(v) Apply left post-inverse theorem twice: $M_1 \twoheadrightarrow M_1^{*\#} \twoheadrightarrow N^\# \twoheadleftarrow M_2^{*\#} \twoheadleftarrow M_2$.

(vi) Apply transitivity of $\twoheadrightarrow_{\Lambda_\$}$ twice: $M_1 \twoheadrightarrow N^\# \twoheadleftarrow M_2$.

□

The following lemma is quite handy in equational reasoning about $\Lambda_\$$ terms.

**Lemma 2.2 (Generalised $=_{\mathbf{bind}}$)** *If $x$ is fresh wrt. $J$, then* $J[M] =_{\Lambda_\$} \text{let}\, x = M \,\text{in}\, J[x]$.

$$
\begin{array}{ll}
\multicolumn{2}{c}{\iota : \lambda_{\$} \to \Lambda_{\$}} \\
\iota(x) & :\equiv x \\
\iota(\lambda\, x\, .\, e) & :\equiv \lambda\, x\, .\, \iota(e) \\
\iota(e\, e') & :\equiv \iota(e)\, \iota(e') \\
\iota(\mathcal{S}_0\, x\, .\, e) & :\equiv \mathcal{S}_0(\lambda\, x\, .\, \iota(e)) \\
\iota(e\, \$\, e') & :\equiv \$(\iota(e'))\, \iota(e)
\end{array}
\qquad
\begin{array}{ll}
\multicolumn{2}{c}{\pi : \Lambda_{\$} \to \lambda_{\$}} \\
\pi(x) & :\equiv x \\
\pi(\lambda\, x\, .\, M) & :\equiv \lambda\, x\, .\, \pi(M) \\
\pi(\$(M)) & :\equiv \lambda\, x\, .\, x\, \$\, \pi(M) \\
\pi(M\, N) & :\equiv \pi(M)\, \pi(N) \\
\pi(\mathcal{S}_0(M)) & :\equiv (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\, \pi(M)
\end{array}
$$

Fig. 5. Embedding $\iota$ and its inverse $\pi$, with macros unfolded.

**Proof.** By a chain of rewrites.

$$
\begin{array}{lr}
 & J[M] \\
\leftarrow_{\mathcal{S}_0\, \$} & J[\mathcal{S}_0(\$(M))] \\
\rightarrow_{\mathbf{bind}} & \text{let } x = \mathcal{S}_0(\$(M)) \text{ in } J[x] \\
\rightarrow_{\mathcal{S}_0\, \$} & \text{let } x = M \text{ in } J[x]
\end{array}
$$

□

## 3 Equational correspondence with $\lambda_{\$}$

In this section, we defend the thesis that $\Lambda_{\$}$ is a calculus of delimited control and its control primitives behave like control operator `shift0` and delimiter `$`. To meet these ends, we show that there exists an equational correspondence with $\lambda_{\$}$ of Materzok [11] expressed with macro-translations. As we mentioned already, $\lambda_{\$}$ has small-step operational semantics, which are sound (but not complete) with respect to axioms and CPS. These semantics allow `shift0` to capture an arbitrarily long context in a single reduction step. Equational correspondence that we prove in this section shows that this traditional approach can also be used in equational reasoning about $\Lambda_{\$}$ terms.

One might wonder, why we only prove equational correspondence and not a Galois connection? We clearly cannot connect directed $\twoheadrightarrow_{\Lambda_{\$}}$ with $=_{\lambda_{\$}}$ because such a connection would transfer symmetry property from $=_{\lambda_{\$}}$ to $\twoheadrightarrow_{\Lambda_{\$}}$, a contradiction: $I\, I \to I \not\twoheadrightarrow I\, I$. We cannot connect $\twoheadrightarrow_{\Lambda_{\$}}$ with $\twoheadrightarrow_{\lambda_{\$}}$ using macros: every macro-translation would necessarily transfer $\lambda\, y\, .\, x\, y \twoheadrightarrow x$ verbatim which is provably true in $\Lambda_{\$}$ and false in $\lambda_{\$}$. Our equational correspondence crucially uses Materzok's $\$_E$ axiom in both directions, so it is unclear how these axioms could be turned into directed reduction rules that would admit a Galois connection. We did not investigate translations that are not based on macros: such translation would weaken our claim that $\Lambda_{\$}$ is a calculus with `shift0`-style delimited control.

Basic reductions of $\Lambda_{\$}$ can be seen as set of axioms. It is later shown in Section 4 as corollary that for such axioms CPS translation $*$ is sound and *complete* ($M =_{\Lambda_{\$}} N$ iff $M^* =_{\lambda} N^*$), which is an improvement upon $\lambda_{\$}$, whose reductions induce an equivalence relation on $\lambda_{\$}$ that is a proper subset of $=_{\lambda_{\$}}$, hence incomplete. These stronger properties make $\Lambda_{\$}$ a contender for the title of the definitive calculus of `shift0`-style delimited control.

To mediate the correspondence, we need translations to ($\iota$) and from ($\pi$) $\Lambda_{\$}$. Translations (with unfolded macros) are presented in Figure 5. With folded macro-definitions they would look like an identity function. To translate from $\lambda_{\$}$ to $\Lambda_{\$}$, we use macros from Figure 3. To translate from $\Lambda_{\$}$ to $\lambda_{\$}$, we use macros from Figure 2.

**Lemma 3.1 (Embedding $\iota$ is invertible)** *The following equalities hold:*

- Left inverse property. *For all $e \in \lambda_{\$}$, $\pi(\iota(e)) = e$.*
- Right inverse property. *For all $M \in \Lambda_{\$}$, $\iota(\pi(M)) = M$.*

**Proof.** Both propositions are separately proven by structural induction.

- Case $e \equiv x$. $\pi(\iota(x)) \equiv \pi(x) \equiv x$.
- Case $e \equiv \lambda\, x\, .\, e_1$. $\pi(\iota(\lambda\, x\, .\, e_1)) \equiv \pi(\lambda\, x\, .\, \iota(e_1)) \equiv \lambda\, x\, .\, \pi(\iota(e_1)) =_{\text{IH}} \lambda\, x\, .\, e_1$.
- Case $e \equiv e_1\, e_2$. $\pi(\iota(e_1\, e_2)) \equiv \pi(\iota(e_1)\, \iota(e_2)) \equiv \pi(\iota(e_1))\, \pi(\iota(e_2)) =_{\text{IH}} e_1\, e_2$.
- Case $e \equiv \mathcal{S}_0\, x\, .\, e_1$.

$$
\begin{array}{lr}
 & \pi(\iota(\mathcal{S}_0\, k\, .\, e_1)) \\
\equiv & \pi(\mathcal{S}_0\, k\, .\, \iota(e_1)) \\
\equiv & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\, \pi(\lambda\, k\, .\, \iota(e_1)) \\
\equiv & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\ \lambda\, k\, .\, \pi(\iota(e_1)) \\
=_{\text{IH}} & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\ \lambda\, k\, .\, e_1 \\
=_{\beta_v} & \mathcal{S}_0\, k\, .(\lambda\, k\, .\, e_1)\, k \\
=_{\beta_v} & \mathcal{S}_0\, k\, .\, e_1
\end{array}
$$

- Case $e \equiv e_1 \,\$\, e_2$.

$$
\begin{array}{lr}
 & \pi(\iota(e_1 \,\$\, e_2)) \\
\equiv & \pi(\iota(e_1) \,\$\, \iota(e_2)) \\
\equiv & \pi(\$(\iota(e_2)))\, \pi(\iota(e_1)) \\
\equiv & (\lambda\, x\, .\, x \,\$\, \pi(\iota(e_2)))\, \pi(\iota(e_1)) \\
=_{\text{IH}} & (\lambda\, x\, .\, x \,\$\, e_2)\, e_1 \\
=_{\eta_\$} & \mathcal{S}_0\, k\, .\, k \,\$\, (\lambda\, x\, .\, x \,\$\, e_2)\, e_1 \\
=_{\$_E} & \mathcal{S}_0\, k\, .(\lambda\, x\, .\, k \,\$\, (\lambda\, x\, .\, x \,\$\, e_2)\, x) \,\$\, e_1 \\
=_{\beta_v} & \mathcal{S}_0\, k\, .(\lambda\, x\, .\, k \,\$\, x \,\$\, e_2) \,\$\, e_1 \\
=_{\$_E} & \mathcal{S}_0\, k\, .\, k \,\$\, e_1 \,\$\, e_2 \\
=_{\eta_\$} & e_1 \,\$\, e_2
\end{array}
$$

- Case $M \equiv x$. $\iota(\pi(x)) \equiv \iota(x) \equiv x$.
- Case $M \equiv \lambda\, x\, .\, M_1$. $\iota(\pi(\lambda\, x\, .\, M_1)) \equiv \iota(\lambda\, x\, .\, \pi(M_1)) \equiv \lambda\, x\, .\, \iota(\pi(M_1)) =_{\text{IH}} \lambda\, x\, .\, M_1$.
- Case $M \equiv \$(M_1)$. $\iota(\pi(\$(M_1))) \equiv \iota(\lambda\, x\, .\, x \,\$\, \pi(M_1)) \equiv \lambda\, x\, .\, \iota(x \,\$\, \pi(M_1)) \equiv \lambda\, x\, .\, \iota(x) \,\$\, \iota(\pi(M_1)) \equiv \lambda\, x\, .\, x \,\$\, \iota(\pi(M_1)) =_{\text{IH}} \lambda\, x\, .\, x \,\$\, M_1 =_{\eta_v} \$(M_1)$.
- Case $M \equiv M_1 M_2$. $\iota(\pi(M_1\, M_2)) \equiv \iota(\pi(M_1)\, \pi(M_2)) \equiv \iota(\pi(M_1))\, \iota(\pi(M_2)) =_{\text{IH}} M_1\, M_2$.
- Case $M \equiv \mathcal{S}_0(M_1)$.

$$
\begin{array}{lr}
 & \iota(\pi(\mathcal{S}_0(M_1))) \\
\equiv & \iota((\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\, \pi(M_1)) \\
\equiv & \iota((\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k))\, \iota(\pi(M_1)) \\
\equiv & (\lambda\, x\, .\, \iota(\mathcal{S}_0\, k\, .\, x\, k))\, \iota(\pi(M_1)) \\
\equiv & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, \iota(x\, k))\, \iota(\pi(M_1)) \\
\equiv & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, \iota(x)\, \iota(k))\, \iota(\pi(M_1)) \\
\equiv & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\, \iota(\pi(M_1))
\end{array}
$$

$$
\begin{array}{ll}
=_{\text{IH}} & (\lambda\, x\, .\, \mathcal{S}_0\, k\, .\, x\, k)\, M_1 \\
=_{\eta_v} & (\lambda\, x\, .\, \mathcal{S}_0(x))\, M_1 \\
=_{\textbf{bind}} & \text{let}\, x = M_1\, \text{in}\, (\lambda\, x\, .\, \mathcal{S}_0(x))\, x \\
=_{\beta_v} & \text{let}\, x = M_1\, \text{in}\, \mathcal{S}_0(x) \\
=_{\textbf{bind}} & \mathcal{S}_0(M_1)
\end{array}
$$

□

**Lemma 3.2** *Equality* $V\,\$\,J[M] = (\lambda\, x\, .\, V\,\$\,J[x])\,\$\,M$ *holds in* $\Lambda_\$$.

**Proof.** By a chain of rewrites.

$$
\begin{array}{ll}
 & V\,\$\,J[M] \\
=_{\textbf{bind}} & V\,\$\,\mathcal{S}_0\, k\, .(\lambda\, y\, .\, k\,\$\,J[y])\,\$\,M \\
=_{\$\,\mathcal{S}_0} & (\lambda\, k\, .(\lambda\, y\, .\, k\,\$\,J[y])\,\$\,M)\, V \\
=_{\beta_v} & (\lambda\, y\, .\, V\,\$\,J[y])\,\$\,M
\end{array}
$$

□

**Lemma 3.3** *Equality* $V\,\$\,K[M] = _{\Lambda_\$}(\lambda\, x\, .\, V\,\$\,K[x])\,\$\,M$ *holds if $x$ is fresh wrt. $V$ and $K$.*

**Proof.** By induction on $K$. In the context of this proof, we use the rules of $\Lambda_\$$ (Fig. 3).

- Base case. $V\,\$\,M =_{\eta_v} (\lambda\, x\, .\, V\, x)\,\$\,M =_{\beta_v} (\lambda\, x\, .(\lambda\, y\, .\, y\, x)\, V)\,\$\,M =_{\$_v} (\lambda\, x\, .\, V\,\$\,x)\,\$\,M$.
- Inductive case. Use of an inductive hypothesis is marked by "IH".

$$
\begin{array}{ll}
 & V\,\$\,J[K[M]] \\
=_{\text{Lemma 3.2}} & (\lambda\, x\, .\, V\,\$\,J[x])\,\$\,K[M] \\
=_{\text{IH}} & (\lambda\, y\, .(\lambda\, x\, .\, V\,\$\,J[x])\,\$\,K[y])\,\$\,M \\
=_{\text{Lemma 3.2}} & (\lambda\, y\, .\, V\,\$\,J[K[y]])\,\$\,M
\end{array}
$$

□

**Theorem 3.4 (Soundness of $\iota$)** *The following propositions hold:*

- *For all $e, e' \in \lambda_\$$, if $e =_{\lambda_\$} e'$, then $\iota(e) =_{\Lambda_\$} \iota(e')$.*
- *For all $M, M' \in \Lambda_\$$, if $\pi(M) =_{\lambda_\$} \pi(M')$, then $M =_{\Lambda_\$} M'$.*

**Proof.** To prove the first proposition it suffices to show that all axioms of $\lambda_\$$ are also valid equalities in $\Lambda_\$$.

- Axiom $\beta_v$ follows immediately by rule $\beta_v$.
- Axiom $\eta_v$ follows immediately by rule $\eta_v$.
- Axiom $\$_v$. $V\,\$\,W \to_{\$_v} (\lambda\, k\, .\, k\, W)\, V \to_{\beta_v} V\, W$.
- Axiom $\$_E$ follows by Lemma 3.3.
- Axiom $\beta_\$$. $V\,\$\,\mathcal{S}_0\, x\, .\, M \to_{\$\,\mathcal{S}_0} (\lambda\, x\, .\, M)\, V \to_{\beta_v} M[V/x]$.
- Axiom $\eta_\$$. $\mathcal{S}_0\, x\, .\, x\,\$\,M \to_{\eta_v} \mathcal{S}_0(\$(M)) \to_{\mathcal{S}_0\,\$} M$.

To prove the second proposition, apply the first proposition on assumption $\pi(M) = \pi(M')$ to get $\iota(\pi(M)) = \iota(\pi(M'))$ and rewrite by the right inverse property of $\iota$ to infer $M = M'$. □

Having the soundness of $\iota$ secured, we move on to completeness. In order to prove it, we show that CPS translation of Materzok is equivalent to ours. Syntax and small-step operational semantics of $\lambda$-calculus are provided for reference in Figure 6.

$$\begin{array}{lll} \text{term} & M, N & ::= x \mid \lambda x\,.\,M \mid M\,N \\ \text{reduction context } C & & ::= [\,] \mid \lambda x\,.\,C \mid C\,M \mid M\,C \end{array}$$

$$\begin{array}{llll} (\lambda x\,.\,M)\,N & \beta & M[N/x] & \\ \lambda x\,.\,M\,x & \eta & M & \text{if } x \text{ is fresh wrt. } M \\ M & \mapsto & N & \text{if } M \beta N \text{ or } M \eta N \\ C[M] & \to & C[N] & \text{if } M \mapsto N \\ M & \twoheadrightarrow & N & \text{if } M \to^n N, n \geq 0 \\ M & = & N & \text{if } M (\to \cup \leftarrow)^n N, n \geq 0 \end{array}$$

Fig. 6. Lambda calculus $\lambda$, the image of CPS translation.

**Lemma 3.5** *Equality* $(M\,N)^* = \lambda k\,.\,M^*\,(\lambda x\,.\,N^*\,(\lambda y\,.\,x\,y\,k))$ *holds in* $\lambda$.

**Proof.** In the appendix. □

**Lemma 3.6 ($\beta\eta$-equivalence of CPS transforms)** *For all* $e \in \lambda_\$$, $\iota(e)^* =_\lambda \mathcal{C}[\![e]\!]$.

**Proof.** By structural induction.

- Case $e \equiv x$. $\iota(x)^* \equiv x^* \equiv \lambda k\,.\,k\,x^\dagger \equiv \lambda k\,.\,k\,x \equiv \mathcal{C}[\![x]\!]$.
- Case $e \equiv \lambda x\,.\,e_1$.

$$\begin{aligned} \iota(\lambda x\,.\,e_1)^* &\equiv (\lambda x\,.\,\iota(e_1))^* \equiv \lambda k\,.\,k\,(\lambda x\,.\,\iota(e_1))^\dagger \equiv \lambda k\,.\,k\ \lambda x\,.\,\iota(e_1)^* \\ &=_{\text{IH}} \lambda k\,.\,k\ \lambda x\,.\,\mathcal{C}[\![e_1]\!] \equiv \lambda k\,.\,k\,\mathcal{C}_v[\![\lambda x\,.\,e_1]\!] \equiv \mathcal{C}[\![\lambda x\,.\,e_1]\!] \end{aligned}$$

- Case $e \equiv e_1\,e_2$.

$$\begin{aligned} \iota(e_1\,e_2)^* &\equiv (\iota(e_1)\,\iota(e_2))^* =_{\text{Lemma 3.5}} \lambda k\,.\,\iota(e_1)^*\,(\lambda x\,.\,\iota(e_2)^*\,(\lambda y\,.\,x\,y\,k)) \\ &=_{\text{IH}} \lambda k\,.\,\mathcal{C}[\![e_1]\!]\,(\lambda x\,.\,\mathcal{C}[\![e_2]\!]\,(\lambda y\,.\,x\,y\,k)) \equiv \mathcal{C}[\![e_1\,e_2]\!] \end{aligned}$$

- Case $e \equiv \mathcal{S}_0\,x\,.\,e_1$.

$$\begin{array}{lr} & \iota(\mathcal{S}_0\,x\,.\,e_1)^* \\ \equiv & \lambda x\,.\,\iota(e_1)^* \\ =_{\text{IH}} & \lambda x\,.\,\mathcal{C}[\![e_1]\!] \\ \equiv & \mathcal{C}[\![\mathcal{S}_0\,x\,.\,e_1]\!] \end{array}$$

- Case $e \equiv e_1\,\$\,e_2$.

$$\begin{array}{lr} & \iota(e_1\,\$\,e_2)^* \\ \equiv & (\iota(e_1)\,\$\,\iota(e_2))^* \\ \equiv & (\$(\iota(e_2))\,\iota(e_1))^* \\ =_{\text{Lemma 3.5}} & \lambda k\,.\,\$(\iota(e_2))^*\,(\lambda x\,.\,\iota(e_1)^*\,(\lambda y\,.\,x\,y\,k)) \\ =_\beta & \lambda k\,.(\lambda x\,.\,\iota(e_1)^*\,(\lambda y\,.\,x\,y\,k))\ \$(\iota(e_2))^\dagger \end{array}$$

$$\# : \lambda \to \Lambda_\$$$

$$\begin{array}{lll} x^\# & :\equiv \mathcal{S}_0(x) & \\ (\lambda\, x\,.\, x\, N)^\# & :\equiv N^\natural & \text{if } x \text{ is fresh wrt. } N \\ (\lambda\, x\,.\, M)^\# & :\equiv \mathcal{S}_0\, x\,.\, M^\# & \text{otherwise} \\ (M\, N)^\# & :\equiv M^\natural\, N^\natural & \end{array} \qquad \begin{array}{ll} x^\natural & :\equiv x \\ (\lambda\, x\,.\, M)^\natural & :\equiv \lambda\, x\,.\, M^\# \\ (M N)^\natural & :\equiv \$(M^\natural\, N^\natural) \end{array}$$

Fig. 7. Conversion from $\lambda$ back to direct style $\Lambda_\$$.

$$\begin{array}{lr} =_\beta & \lambda\, k\,.\, \iota(e_1)^*\, (\lambda\, y\,.\, \$(\iota(e_2))^\dagger\, y\, k) \\ \equiv & \lambda\, k\,.\, \iota(e_1)^*\, (\lambda\, y\,.\, \iota(e_2)^*\, y\, k) \\ =_{\text{IH}} & \lambda\, k\,.\, \mathcal{C}[\![e_1]\!]\, (\lambda\, y\,.\, \mathcal{C}[\![e_2]\!]\, y\, k) \\ \equiv & \mathcal{C}[\![e_1\, \$\, e_2]\!] \end{array}$$

□

We strike the final nail in the coffin of doubt with the completeness theorem.

**Theorem 3.7 (Completeness of $\iota$)** *The following propositions hold:*

- *For all $e, e' \in \lambda_\$$, if $\iota(e) =_{\Lambda_\$} \iota(e')$, then $e =_{\lambda_\$} e'$.*
- *For all $M, M' \in \Lambda_\$$, if $M =_{\Lambda_\$} M'$, then $\pi(M) =_{\lambda_\$} \pi(M')$.*

**Proof.** To prove the first proposition, apply monotonicity of $*$ (Theorem 4.8) on assumption $\iota(e) = \iota(e')$ to get $\iota(e)^* = \iota(e')^*$. Rewrite using equivalence of CPS translations to get $\mathcal{C}[\![e]\!] = \mathcal{C}[\![e']\!]$ and finally apply completeness of $\mathcal{C}[\![\cdot]\!]$ to conclude that $e = e'$. To prove the second proposition, rewrite assumption $M = M'$ with left inverse property of $\iota$ to get $\iota(\pi(M)) = \iota(\pi(M))$. Apply the first proposition to finally arrive at $\pi(M) = \pi(M')$. □

It follows that calculi $\Lambda_\$$ and $\lambda_\$$ correspond equationally via macro-definitions, hence our $\Lambda_\$$ is truly a calculus of `shift0`-style delimited control.

## 4 Reflection of lambda calculus into $\Lambda_\$$

We move on to the main result of this paper: the relationship between $\Lambda_\$$ and $\lambda$. It will be shown that $\lambda$ reflects into $\Lambda_\$$. Syntax and small-step operational semantics of $\lambda$-calculus are provided for reference in Figure 6. Following in the footsteps of Sabry and Wadler [18] we intended to define a backwards, *direct style* translation $\#$ from the range of $*$, just like in [1] and [2]. We took the range of the old CPS and closed it under reduction, following the recipe by Sabry and Felleisen [17]. What we've got was the entire set of lambda terms. A question arised: was there a better CPS that would explicitly hit every possible lambda term? The answer is positive. In other words, for *every* term $M \in \lambda$, there exists a term $M^\# \in \Lambda_\$$, such that $M^{\#*} \equiv M$. Every lambda term is in Continuation-Passing Style, it seems!

Translation $\#$ has one oddity: special treatment of $\lambda\, x\,.\, x\, N$ when $x$ is fresh wrt. $N$. One may notice that it is the shape of a value translated to CPS. This is not essential for the right inverse theorem to hold but it is a necessary adjustment for the left post-inverse theorem.

When it comes to discovery of those translations, it was mostly trial and error, working within confines of a reflection, fine-tuning simultaneously the translations and the calculus. Just as CPS translations come in a pair: one on value, one on general terms, both had to be inverted and therefore we have in fact two different embeddings of lambda terms: one embeds into values of $\Lambda_\$$ and the other into general terms of $\Lambda_\$$. They need each other to compute DS translation just like the two CPS translations interleave to bring a CPS term.

**Example 4.1** Let's take the CPS term from the previous example and reflect it into $\Lambda_\$$.

$$(\lambda\, k_1 \,.\, k_1\, (\lambda\, x\, k_2 \,.\, k_2\, x))^{\#} \equiv (\lambda\, x\, k_2 \,.\, k_2\, x)^{\natural} \equiv \lambda\, x \,.(\lambda\, k_2 \,.\, k_2\, x)^{\#} \equiv \lambda\, x \,.\, x^{\natural} \equiv \lambda\, x \,.\, x.$$

**Theorem 4.2 (Right inverse of $*$)** *For all $M \in \lambda$, $M^{\#*} \equiv M$ and $M^{\natural\dagger} \equiv M$.*

**Proof.** Structural induction on $M$.

- Case $M \equiv x$. $x^{\#*} \equiv \mathcal{S}_0(x)^* \equiv x^\dagger \equiv x$ and $x^{\natural\dagger} \equiv x^\dagger \equiv x$.
- Case $M \equiv \lambda\, x \,.\, M_1$. First identity has two subcases.
  · Case $M_1 \equiv x\, M_2$ and $x$ is fresh wrt. $M_2$. $(\lambda\, x \,.\, x\, M_2)^{\#*} \equiv M_2^{\natural *} \equiv \lambda\, x \,.\, x\, M_2^{\natural\dagger} \equiv_{\mathrm{IH}} \lambda\, x \,.\, x\, M_2$.
  · Opposite case. $(\lambda\, x \,.\, M_1)^{\#*} \equiv (\mathcal{S}_0\, x \,.\, M_1^{\#})^* \equiv (\lambda\, x \,.\, M_1^{\#})^\dagger \equiv \lambda\, x \,.\, M_1^{\#*} \equiv_{\mathrm{IH}} \lambda\, x \,.\, M_1$.
  Second identity: $(\lambda\, x \,.\, M_1)^{\natural\dagger} \equiv (\lambda\, x \,.\, M_1^{\#})^\dagger \equiv \lambda\, x \,.\, M_1^{\#*} \equiv_{\mathrm{IH}} \lambda\, x \,.\, M_1$.
- Case $M \equiv M_1\, M_2$. $(M_1\, M_2)^{\#*} \equiv (M_1^{\natural}\, M_2^{\natural})^* \equiv M_1^{\natural\dagger}\, M_2^{\natural\dagger} \equiv_{\mathrm{IH}} M_1\, M_2$, $(M_1\, M_2)^{\natural\dagger} \equiv \$(M_1^{\natural}\, M_2^{\natural})^\dagger \equiv (M_1^{\natural}\, M_2^{\natural})^* \equiv M_1^{\natural\dagger}\, M_2^{\natural\dagger} \equiv_{\mathrm{IH}} M_1\, M_2$.

□

### *4.1 Monotonicity of CPS and DS translation*

In this subsection we establish that both $*$ and $\#$ are monotone. A few lemmas are needed for these to work, notably substitution lemmas. Some lemmas could be proven by structural induction, but it is more concise and natural for them to follow recursion pattern of the definition of $*$. For this reason, we prove these lemmas by induction on the size of $M$. Since we have a grammar of valid terms, we simply define the size of a term as the number of nodes in a parse tree of that term. Crucially, variables are smaller in this sense than nonvalues, hence $J[x]$ is always smaller than $J[P]$.

**Lemma 4.3 (Values reduce to values)** *If $V \to_{\Lambda_\$} N$, then for some $W$, $N \equiv W$.*

**Proof.** By definition of $\to$, there exist $M'$, $N'$ and $C$, such that, $M' \mapsto N'$, $V \equiv C[M']$ and $N \equiv C[N']$. We proceed by cases on $V$ and $C$.

- Case $V \equiv x$. Necessarily $C \equiv [\,]$. There is no possible reduction, proposition holds vacuously.
- Case $V \equiv \lambda\, x \,.\, M_1$.
  · Subcase $C \equiv [\,]$. Necessarily $V(\eta_v)N$. By definition of $\eta_v$, $V \equiv \lambda\, x \,.\, x\, W$ and $N \equiv W$ for some $W$.
  · Subcase $C \equiv \lambda\, x \,.\, C'$. $N \equiv C[N'] \equiv \lambda\, x \,.\, C'[N']$.
- Case $V \equiv \$(M_1)$.
  · Subcase $C \equiv [\,]$ and $V(\$_v)N$. By definition of $\$_v$, $V \equiv \$(W)$ and $N \equiv \lambda\, x \,.\, x\, W$ for some $W$.
  · Subcase $C \equiv [\,]$ and $V(\$\,\mathcal{S}_0)N$. By definition of $\$\,\mathcal{S}_0$, $V \equiv \$(\mathcal{S}_0(W))$ and $N \equiv W$ for some $W$.
  · Subcase $C \equiv \$(C')$. $N \equiv C[N'] \equiv \$(C'[N'])$.

□

**Lemma 4.4** • *If $V^* \to_\lambda W^*$, then $V^\dagger \to_\lambda W^\dagger$.*

**Proof.** Assume $V^* \to W^*$. Unfolding the definition of $*$ gives $\lambda\, k \,.\, k\, V^\dagger \to \lambda\, k \,.\, k\, W^\dagger$. By definition of $\to$ (in $\lambda$), there exist $M$, $N$ and $C$, such that, $M(\beta \cup \eta)N$, $\lambda\, k \,.\, k\, V^\dagger \equiv C[M]$ and $\lambda\, k \,.\, k\, W^\dagger \equiv C[N]$. We proceed by cases on $C$.

- Case $C \equiv [\,]$ or $C \equiv \lambda\, k \,.[\,]$. There is no possible reduction, proposition holds vacuously.
- Case $C \equiv \lambda\, k \,.\, k\, C'$. Given that $M(\beta \cup \eta)N$, $V^\dagger \equiv C'[M]$ and $W^\dagger \equiv C'[N]$, we infer $V^\dagger \to W^\dagger$.

□

**Lemma 4.5 (Substitution lemma for $*$ and $\dagger$)** *For every $M$ and $V$,*

- $M^*[V^\dagger/x] \equiv M[V/x]^*$ *and*

- *if $M$ is a value, then $M^\dagger[V^\dagger/x] \equiv M[V/x]^\dagger$.*

**Proof.** By induction on the size of $M$. We prove the second conjunct below.

- Case $M \equiv x$. $x^\dagger[V^\dagger/x] \equiv x[V^\dagger/x] \equiv V^\dagger \equiv x[V/x]^\dagger$
- Case $M \equiv y$. $y^\dagger[V^\dagger/x] \equiv y[V^\dagger/x] \equiv y \equiv y^\dagger \equiv y[V/x]^\dagger$.
- Case $M \equiv \lambda\, y\,.\, M_1$. $(\lambda\, y\,.\, M_1)^\dagger[V^\dagger/x] \equiv \lambda\, y\,.\, M_1^*[V^\dagger/x] \equiv_{\text{IH}} \lambda\, y\,.\, M_1[V/x]^* \equiv (\lambda\, y\,.\, M_1)[V/x]^\dagger$.
- Case $M \equiv \$(M_1)$. $\$(M_1)^\dagger[V^\dagger/x] \equiv M_1^*[V^\dagger/x] \equiv_{\text{IH}} M_1[V/x]^* \equiv \$(M_1)[V/x]^\dagger$.

We now prove the first conjunct.

- Case $M \equiv J[P]$.

$$J[P]^*[V^\dagger/x] \equiv \lambda\, k\,.\, P^*[V^\dagger/x]\,(\lambda\, y\,.\, J[y]^*[V^\dagger/x]\, k) \equiv_{\text{IH}} \lambda\, k\,.\, P[V/x]^*\,(\lambda\, y\,.\, J[y][V/x]^*\, k) \equiv J[P][V/x]^*$$

- Case $M \equiv V_1\, V_2$. $(V_1\, V_2)^*[V^\dagger/x] \equiv V_1^\dagger[V^\dagger/x]\, V_2^\dagger[V^\dagger/x] \equiv_{\text{IH}} V_1[V/x]^\dagger\, V_2[V/x]^\dagger \equiv (V_1\, V_2)[V/x]^*$
- Case $M \equiv \mathcal{S}_0(V_1)$. $\mathcal{S}_0(V_1)^*[V^\dagger/x] \equiv V_1^\dagger[V^\dagger/x] \equiv_{\text{IH}} V_1[V/x]^\dagger \equiv \mathcal{S}_0(V_1)[V/x]^*$
- Case $M \equiv W$. For brevity, we consider all values at once and reuse the proofs of the second conjunct. $W^*[V^\dagger/x] \equiv \lambda\, k\,.\, k\, W^\dagger[V^\dagger/x] \equiv_{\text{the second conjunct}} \lambda\, k\,.\, k\, W[V/x]^\dagger \equiv W[V/x]^*$.

□

**Lemma 4.6** *If $M_1 \mapsto_{\Lambda_\$} M_2$, then $M_1^* \to_\lambda^? M_2^*$.*

**Proof.**

- Case $\beta_v$. $((\lambda\, x\,.\, M)\, V)^* \equiv (\lambda\, x\,.\, M^*)\, V^\dagger \to M^*[V^\dagger/x] \equiv_{\text{Lemma 4.5}} M[V/x]^*$.
- Case $\eta_v$. $(\lambda\, x\,.\, V\, x)^* \equiv \lambda\, k\,.\, k\,(\lambda\, x\,.\, V^\dagger\, x) \to_\eta \lambda\, k\,.\, k\, V^\dagger \equiv V^*$.
- Case $\$_v$. $\$(V)^* \equiv \lambda\, k\,.\, k\ \$(V)^\dagger \equiv \lambda\, k\,.\, k\, V^* \equiv \lambda\, k\,.\, k\,(\lambda\, x\,.\, x\, V^\dagger) \equiv \lambda\, k\,.\, k\,(\lambda\, x\,.\, x\, V)^\dagger \equiv (\lambda\, x\,.\, x\, V)^*$.
- Case $\$\,\mathcal{S}_0$. $\$(\mathcal{S}_0(V))^* \equiv \lambda\, k\,.\, k\ \$(\mathcal{S}_0(V))^\dagger \equiv \lambda\, k\,.\, k\ \mathcal{S}_0(V)^* \equiv \lambda\, k\,.\, k\, V^\dagger \equiv V^*$.
- Case $\mathcal{S}_0\,\$$. $\mathcal{S}_0(\$(M))^* \equiv \$(M)^\dagger \equiv M^*$.
- Case **pure**. $(\mathcal{S}_0\, x\,.\, x\, V)^* \equiv \lambda\, x\,.(x\, V)^* \equiv \lambda\, x\,.\, x\, V^\dagger \equiv V^*$.
- Case **bind**.

$$\begin{aligned} J[P]^* &\equiv \lambda\, k\,.\, P^*\,(\lambda\, x\,.\, J[x]^*\, k) \equiv \lambda\, k\,.\, \$(P)^\dagger\,(\lambda\, x\,.\, \$(J[x])^\dagger\, k) \equiv \lambda\, k\,.\, \$(P)^\dagger\,(\lambda\, x\,.\, k\,\$\, J[x])^\dagger \\ &\equiv (\lambda\, k\,.(\lambda\, x\,.\, k\,\$\, J[x])\,\$\, P)^\dagger \equiv (\mathcal{S}_0\, k\,.(\lambda\, x\,.\, k\,\$\, J[x])\,\$\, P)^* \equiv (\text{let}\, x = P\, \text{in}\, J[x])^* \end{aligned}$$

□

**Lemma 4.7 (Single-step reduction preservation by $*$)** *If $M \to_{\Lambda_\$} N$, then $M^* \to_\lambda^? N^*$.*

**Proof.** By structural induction on $M$. We unfold the definition of $\to$ (in $\Lambda_\$$): there exist $M'$, $N'$ and $C$, such that, $M' \mapsto N'$, $M \equiv C[M']$ and $N \equiv C[N']$. If $M$ is a variable, then there is no possible reduction and the proposition holds vacuously. If $C \equiv [\,]$, then apply Lemma 4.6. Otherwise, we proceed by cases on $M$ and (nonempty) $C$. In each case below we consider some proper subcontext $C'$ of $C$. Because $C'[M']$ is a proper subterm of $M$, then from inductive hypothesis and $C'[M'] \to C'[N']$ we infer $C'[M']^* \to^? C'[N']^*$. When $C'[M']$ is a value, it reduces to a value $C'[N']$, so $C'[N']^\dagger$ is well-defined and Lemma 4.4 gives us $C'[M']^\dagger \to^? C'[N']^\dagger$.

- Case $M \equiv \lambda\, x\,.\, M_1$ and $C \equiv \lambda\, x\,.\, C'$. $(\lambda\, x\,.\, M_1)^* \equiv \lambda\, k\,.\, k\,(\lambda\, x\,.\, M_1^*) \equiv \lambda\, k\,.\, k\,(\lambda\, x\,.\, C'[M']^*) \to_{\text{IH}}^? \lambda\, k\,.\, k\,(\lambda\, x\,.\, C'[N']^*) \equiv (\lambda\, x\,.\, C'[N'])^* \equiv N^*$.
- Case $M \equiv \$(M_1)$ and $C \equiv \$(C')$. $\$(M_1)^* \equiv \lambda\, k\,.\, k\, M_1^*) \equiv \lambda\, k\,.\, k\, C'[M']^* \to_{\text{IH}}^? \lambda\, k\,.\, k\, C'[N']^* \equiv (\$(C'[N']))^* \equiv N^*$.

- Case $M \equiv \mathcal{S}_0(V)$ and $C \equiv \mathcal{S}_0(C')$. $\mathcal{S}_0(V)^* \equiv V^\dagger \equiv C'[M']^\dagger \to^?_{\mathrm{IH}} C'[N']^\dagger \equiv \mathcal{S}_0(C'[N'])^* \equiv N^*$.
- Case $M \equiv J[P]$ and $C \equiv J[C']$. $J[P]^* \equiv \lambda k \,.\, P^* \, (\lambda x \,.\, J[x]^* \, k) \equiv \lambda k \,.\, C'[M']^* \, (\lambda x \,.\, J[x]^* \, k) \to^?_{\mathrm{IH}} \lambda k \,.\, C'[N']^* \, (\lambda x \,.\, J[x]^* \, k) \equiv J[C'[N']]^* \equiv N^*$.
- Case $M \equiv V_1 \, V_2$ and $C \equiv C' \, V_2$. $(V_1 \, V_2)^* \equiv V_1^\dagger \, V_2^\dagger \equiv C'[M']^\dagger \, V_2^\dagger \to^?_{\mathrm{IH}} C'[N']^\dagger \, V_2^\dagger \equiv (C'[N'] \, V_2)^* \equiv N^*$.
- Case $M \equiv V_1 \, V_2$ and $C \equiv V_1 \, C'$. $(V_1 \, V_2)^* \equiv V_1^\dagger \, V_2^\dagger \equiv V_1^\dagger \, C'[M']^\dagger \to^?_{\mathrm{IH}} V_1^\dagger \, C'[N']^\dagger \equiv (V_1 \, C'[N'])^* \equiv N^*$.
- Case $M \equiv V_1 \, P_2$ and $C \equiv C' \, P_2$. $(V_1 \, P_2)^* \equiv \lambda k \,.\, P_2^* \, (\lambda x \,.\, V_1^\dagger \, x \, k) \equiv \lambda k \,.\, P_2^* \, (\lambda x \,.\, C'[M']^\dagger \, x \, k) \to^?_{\mathrm{IH}} \lambda k \,.\, P_2^* \, (\lambda x \,.\, C'[N']^\dagger \, x \, k) \equiv (C'[N'] \, P_2)^* \equiv N^*$.
- Case $M \equiv P_1 \, V_2$ and $C \equiv P_1 \, C'$. $(P_1 \, V_2)^* \equiv \lambda k \,.\, P_1^* \, (\lambda x \,.\, x \, V_2^\dagger \, k) \equiv \lambda k \,.\, P_1^* \, (\lambda x \,.\, x \, C'[M']^\dagger \, k) \to^?_{\mathrm{IH}} \lambda k \,.\, P_1^* \, (\lambda x \,.\, x \, C'[N']^\dagger \, k) \equiv (P_1 \, C'[N'])^* \equiv N^*$.
- Case $M \equiv P_1 \, P_2$ and $C \equiv P_1 \, C'$.

$$\begin{aligned}(P_1 \, P_2)^* &\equiv \lambda k \,.\, P_1^* \, (\lambda x \,.\, \lambda k \,.\, P_2^* \, (\lambda y \,.\, x \, y \, k) \, k) \equiv \lambda k \,.\, P_1^* \, (\lambda x \,.\, \lambda k \,.\, C'[M']^* \, (\lambda y \,.\, x \, y \, k) \, k) \\ &\to^?_{\mathrm{IH}} \lambda k \,.\, P_1^* \, (\lambda x \,.\, \lambda k \,.\, C'[N']^* \, (\lambda y \,.\, x \, y \, k) \, k) \equiv (P_1 \, C'[N'])^* \equiv N^*\end{aligned}$$

□

**Theorem 4.8 (Monotonicity of $*$)** *For all $M, N \in \Lambda_\$$, $M \twoheadrightarrow_{\Lambda_\$} N$ implies $M^* \twoheadrightarrow_\lambda N^*$.*

**Proof.** By induction on the number $n$ of reduction steps in $M \twoheadrightarrow N$.

- Case $n = 0$, $M \equiv N$. $M^* \twoheadrightarrow N^*$ in 0 steps.
- Case $n > 0$, $M \to^{n-1} L \to N$ for some $L$. $M^* \twoheadrightarrow_{\mathrm{IH}} L^* \to^?_{\text{Lemma 4.7}} N^*$.

□

**Lemma 4.9** *For all $M \in \lambda$, $\mathcal{S}_0(M^\natural) \to^?_{\Lambda_\$} M^\#$.*

**Proof.** By cases on $M$.

- Case $M \equiv x$. $\mathcal{S}_0(x^\natural) \equiv \mathcal{S}_0(x) \equiv x^\#$.
- Case $M \equiv \lambda x \,.\, M_1$.
  - · Case $M_1 \equiv x \, M_2$ and $x$ is fresh wrt. $M_2$. $\mathcal{S}_0((\lambda x \,.\, x \, M_2)^\natural) \equiv \mathcal{S}_0 \, x \,.\, x \, M_2^\natural \to_{\mathbf{pure}} M_2^\natural \equiv (\lambda x \,.\, x \, M_2)^\#$.
  - · Opposite case. $\mathcal{S}_0((\lambda x \,.\, M_1)^\natural) \equiv \mathcal{S}_0 \, x \,.\, M_1^\# \equiv (\lambda x \,.\, M_1)^\#$.
- Case $M \equiv M_1 \, M_2$. $\mathcal{S}_0((M_1 \, M_2)^\natural) \equiv \mathcal{S}_0(\$(M_1^\natural \, M_2^\natural)) \to_{\mathcal{S}_0 \, \$} M_1^\natural \, M_2^\natural \equiv (M_1 \, M_2)^\#$.

□

**Lemma 4.10** *For all $M \in \lambda$, $\$(M^\#) \to^?_{\Lambda_\$} M^\natural$.*

**Proof.** By cases on $M$.

- Case $M \equiv x$. $\$(x^\#) \equiv \$(\mathcal{S}_0(x)) \to_{\$ \, \mathcal{S}_0} x \equiv x^\natural$.
- Case $M \equiv \lambda x \,.\, M_1$.
  - · Case $M_1 \equiv x \, M_2$ and $x$ is fresh wrt. $M_2$. $\$((\lambda x \,.\, x \, M_2)^\#) \equiv \$(M_2^\natural) \to_{\$_v} \lambda x \,.\, x \, M_2^\natural \equiv (\lambda x \,.\, x \, M_2)^\natural$.
  - · Opposite case. $\$((\lambda x \,.\, M_1)^\#) \equiv \$(\mathcal{S}_0 \, x \,.\, M_1^\#) \to_{\$ \, \mathcal{S}_0} \lambda x \,.\, M_1^\# \equiv (\lambda x \,.\, M_1)^\natural$.
- Case $M \equiv M_1 \, M_2$. $\$((M_1 \, M_2)^\#) \equiv \$(M_1^\natural \, M_2^\natural) \equiv (M_1 \, M_2)^\natural$.

□

**Lemma 4.11** *$M^\#[N^\natural/x] \twoheadrightarrow_{\Lambda_\$} M[N/x]^\#$ and $M^\natural[N^\natural/x] \twoheadrightarrow_{\Lambda_\$} M[N/x]^\natural$ hold.*

**Proof.** In the appendix. □

**Lemma 4.12** *If $M \beta N$ or $M \eta N$, then*

- *$M^\# \twoheadrightarrow_{\Lambda_\$} N^\#$ and*

- $M^\natural \twoheadrightarrow_{\Lambda_\$} N^\natural$.

**Proof.**

- Case $\beta$.
  - $((\lambda x\,.\,M)\,N)^\# \equiv (\lambda x\,.\,M^\#)\,N^\natural \to_{\beta_v} M^\#[N^\natural/x] \to_{\text{Lemma 4.11}} M[N/x]^\#$.
  - $((\lambda x\,.\,M)\,N)^\natural \equiv \$((\lambda x\,.\,M^\#)\,N^\natural) \to_{\beta_v} \$(M^\#[N^\natural/x]) \to_{\text{Lemma 4.11}} \$(M[N/x]^\#) \to^?_{\text{Lemma 4.10}} M[N/x]^\natural$.
- Case $\eta$.
  - $(\lambda x\,.\,M\,x)^\# \equiv \mathcal{S}_0(\lambda x\,.\,M^\natural\,x) \to_{\eta_v} \mathcal{S}_0(M^\natural) \to^?_{\text{Lemma 4.9}} M^\#$.
  - $(\lambda x\,.\,M\,x)^\natural \equiv \lambda x\,.\,M^\natural\,x \to_{\eta_v} M^\natural$.

□

**Lemma 4.13 (Single-step reduction preservation by # and ♮)** *If $M \to_\lambda N$, then*

- $M^\# \twoheadrightarrow_{\Lambda_\$} N^\#$ *and*
- $M^\natural \twoheadrightarrow_{\Lambda_\$} N^\natural$.

**Proof.** By structural induction on $M$. We unfold the definition of $\to$ (in $\lambda$): there exist $M'$, $N'$ and $C$, such that, $M' \mapsto N'$, $M \equiv C[M']$ and $N \equiv C[N']$. If $M$ is a variable, then there is no possible reduction and the proposition holds vacuously. If $C \equiv [\,]$, then apply Lemma 4.12. Otherwise, we proceed by cases on $M$ and (nonempty) $C$. In each case below we consider some proper subcontext $C'$ of $C$. Because $C'[M']$ is a proper subterm of $M$, then from inductive hypothesis and $C'[M'] \to C'[N']$ we infer $C'[M']^\# \twoheadrightarrow C'[N']^\#$ and $C'[M']^\natural \twoheadrightarrow C'[N']^\natural$.

- Case $M \equiv M_1\,M_2$ and $C \equiv C'\,M_2$.
  - $(M_1\,M_2)^\# \equiv M_1^\natural\,M_2^\natural \equiv C'[M']^\natural\,M_2^\natural \twoheadrightarrow_{\text{IH}} C'[N']^\natural\,M_2^\natural \equiv (C'[N']\,M_2)^\# \equiv N^\#$.
  - $(M_1\,M_2)^\natural \equiv \$(M_1^\natural\,M_2^\natural) \equiv \$(C'[M']^\natural\,M_2^\natural) \twoheadrightarrow_{\text{IH}} \$(C'[N']^\natural\,M_2^\natural) \equiv (C'[N']\,M_2)^\natural \equiv N^\natural$.
- Case $M \equiv M_1\,M_2$ and $C \equiv M_1\,C'$.
  - $(M_1\,M_2)^\# \equiv M_1^\natural\,M_2^\natural \equiv M_1^\natural\,C'[M']^\natural \twoheadrightarrow_{\text{IH}} M_1^\natural\,C'[N']^\natural \equiv (M_1\,C'[N'])^\# \equiv N^\#$.
  - $(M_1\,M_2)^\natural \equiv \$(M_1^\natural\,M_2^\natural) \equiv \$(M_1^\natural\,C'[M']^\natural) \twoheadrightarrow_{\text{IH}} \$(M_1^\natural\,C'[N']^\natural) \equiv (M_1\,C'[N'])^\natural \equiv N^\natural$.
- Case $M \equiv \lambda x\,.\,M_1$ and $C \equiv \lambda x\,.\,C'$, second conjunct. $(\lambda x\,.\,M_1)^\natural \equiv \lambda x\,.\,M_1^\# \equiv \lambda x\,.\,C'[M']^\# \twoheadrightarrow_{\text{IH}} \lambda x\,.\,C'[N']^\# \equiv (\lambda x\,.\,C'[N'])^\natural \equiv N^\natural$.
- Case $M \equiv \lambda x\,.\,M_1$, first conjunct.
  - Case $M_1 \equiv x\,M_2$, $x$ is fresh wrt. $M_2$ and $C \equiv \lambda x\,.\,x\,C'$. $(\lambda x\,.\,x\,M_2)^\# \equiv M_2^\natural \equiv C'[M']^\natural \twoheadrightarrow_{\text{IH}} C'[N']^\natural \equiv (\lambda x\,.\,x\,C'[N'])^\# \equiv N^\#$.
  - Opposite case, $C \equiv \lambda x\,.\,C'$.

$$\begin{aligned}(\lambda x\,.\,M_1)^\# &\equiv \mathcal{S}_0\,x\,.\,M_1^\# \equiv \mathcal{S}_0\,x\,.\,C'[M']^\# \twoheadrightarrow_{\text{IH}} \mathcal{S}_0\,x\,.\,C'[N']^\# \\ &\equiv \mathcal{S}_0((\lambda x\,.\,C'[N'])^\natural) \to^?_{\text{Lemma 4.9}} (\lambda x\,.\,C'[N'])^\#\end{aligned}$$

□

**Theorem 4.14 (Monotonicity of #)** *For all $M, N \in \lambda$, if $M \twoheadrightarrow N$, then $M^\# \twoheadrightarrow N^\#$.*

**Proof.** By induction on the number $n$ of reduction steps in $M \twoheadrightarrow N$.

- Case $n = 0$, $M \equiv N$. $M^* \twoheadrightarrow N^*$ in 0 steps.
- Case $n > 0$, $M \to^{n-1} L \to N$ for some $L$. $M^* \twoheadrightarrow_{\text{IH}} L^* \twoheadrightarrow_{\text{Lemma 4.13}} N^*$.

□

### 4.2 The reflection theorem and the kernel calculus

The last missing ingredient of reflection is the left post-inverse theorem. Inspection of the proof reveals that neither $\beta_v$ nor $\eta_v$ is used while every other rule is used in some instance.

**Theorem 4.15 (Left post-inverse of $*$)** *For all $M \in \Lambda_\$$,*

- $M \twoheadrightarrow M^{*\#}$ *and*
- *if $M$ is a value, then $M \twoheadrightarrow M^{\dagger\natural}$.*

**Proof.** By induction on the size of $M$. We prove the second conjunct below.

- Case $M \equiv x$. $x \equiv x^\natural \equiv x^{\dagger\natural}$.
- Case $M \equiv \lambda x\,.\,M_1$. $\lambda x\,.\,M_1 \twoheadrightarrow_{\mathrm{IH}} \lambda x\,.\,M_1^{*\#} \equiv (\lambda x\,.\,M_1^*)^\natural \equiv (\lambda x\,.\,M_1)^{\dagger\natural}$.
- Case $M \equiv \$(M_1)$. $\$(M_1) \twoheadrightarrow_{\mathrm{IH}} \$(M_1^{*\#}) \to^?_{\text{Lemma 4.10}} M_1^{*\natural} \equiv \$(M_1)^{\dagger\natural}$.

We now prove the first conjunct.

- Case $M \equiv J[P]$.

$$J[P] \to_{\mathbf{bind}} \mathrm{let}\, x = P \,\mathrm{in}\, J[x] \twoheadrightarrow_{\mathrm{IH}} \mathrm{let}\, x = P^{*\#} \,\mathrm{in}\, J[x]^{*\#} \equiv \mathcal{S}_0\, k\,.\,\$(P^{*\#})\,(\lambda x\,.\,\$(J[x]^{*\#})\,k)$$
$$\to^?_{\text{Lemma 4.10}} \mathcal{S}_0\, k\,.\,P^{*\natural}\,(\lambda x\,.\,J[x]^{*\natural}\,k) \equiv (\lambda k\,.\,P^*\,(\lambda x\,.\,J[x]^*\,k))^\# \equiv J[P]^{*\#}$$

- Case $M \equiv V_1\,V_2$. $V_1\,V_2 \twoheadrightarrow_{\mathrm{IH}} V_1^{\dagger\natural}\,V_2^{\dagger\natural} \equiv (V_1^\dagger\,V_2^\dagger)^\# \equiv (V_1\,V_2)^{*\#}$.
- Case $M \equiv \mathcal{S}_0(V_1)$. $\mathcal{S}_0(V_1) \twoheadrightarrow_{\mathrm{IH}} \mathcal{S}_0(V_1^{\dagger\natural}) \to^?_{\text{Lemma 4.9}} V_1^{\dagger\#} \equiv \mathcal{S}_0(V_1)^{*\#}$.
- Case $M \equiv W$. For brevity, we consider all values at once and reuse the proofs of the second conjunct. $W \twoheadrightarrow_{\text{the second conjunct}} W^{\dagger\natural} \equiv (\lambda k\,.\,k\,W^\dagger)^\# \equiv W^{*\#}$.

□

**Example 4.16** The CPS translation of $S$-combinator $\lambda x\,y\,z\,.\,x\,z\,(y\,z)$ is

$$S^* \equiv \lambda k_1\,.\,k_1\,(\lambda x\,k_2\,.\,k_2\,(\lambda y\,k_3\,.\,k_3\,(\lambda z\,k_4\,.\,x\,y\,(\lambda f\,.(\lambda k_5\,.\,y\,z\,(\lambda a\,.\,f\,a\,k_5))\,k_4)))).$$

Reflection of this CPS term back to $\Lambda_\$$ is $S^{*\#} \equiv \lambda x\,y\,z\,.\,\mathcal{S}_0\,k_4\,.(\lambda f\,.(\lambda k_5\,.(\lambda a\,.\,k_5\,\$\,f\,a)\,\$\,y\,z)\,k_4)\,\$\,x\,y$. One can see that $S \twoheadrightarrow_{\mathbf{bind}} \lambda x\,y\,z\,.\,\mathrm{let}\, f = x\,y \,\mathrm{in}\, \mathrm{let}\, a = y\,z \,\mathrm{in}\, f\,a \to_{\$\,\mathcal{S}_0} S^{*\#}$, in accordance with Theorem 4.15.

**Theorem 4.17 ($(*, \#)$ form a Galois connection)** *For any $M \in \Lambda_\$$ and $N \in \lambda$ we have $M \twoheadrightarrow_{\Lambda_\$} N^\#$ if and only if $M^* \twoheadrightarrow_\lambda N$.*

**Proof.** The left-to-right implication follows from the monotonicity of $*$ (Theorem 4.8) and the right inverse property (Theorem 4.2). The right-to-left implication follows from the monotonicity of $\#$ (Theorem 4.14). and the left post-inverse property (Theorem 4.15). □

**Corollary 4.18 (Reflection)** *Since $\#$ is a right inverse of $*$, the Galois connection is indeed a reflection.*

The range of $\#$ translation (from now on, the *kernel*, or $\lambda^\#$) has an induced preorder

$$\twoheadrightarrow_{\lambda^\#} = \left\{ (M, N) \in \lambda^\# \times \lambda^\# \middle| M \twoheadrightarrow_{\Lambda_\$} N \right\}.$$

The kernel can be considered a calculus in its own right and it turns out we know this calculus very well: it is preorder-isomorphic to $\lambda$ via $*$ restricted to kernel and its inverse $\#$. What that means is that the kernel is a fragment of $\Lambda_\$$ which behaves exactly like $\lambda$-calculus ($M \twoheadrightarrow_{\lambda^\#} N$ iff $M^* \twoheadrightarrow_\lambda N^*$) except that terms look different and $*$ allows to decode these to familiar $\lambda$-terms. We believe that reductions in kernel calculus could be fully characterized, in similar fashion to [18,1,2], but we did not investigate that yet. Experience suggest that these reductions would be quite verbose.

**Corollary 4.19** *Calculi $\Lambda\mu_v$ [4], $\lambda_\$$ [11], $\Lambda_\$$ and $\lambda$ are in equational correspondence.*

**Proof.** Calculus $\Lambda\mu_v$ is a calculus created by relaxing some syntactic restrictions of $\lambda\mu_v$. $\Lambda\mu_v$ was defined and proven to correspond equationally with $\lambda_\$$ in [4]. Calculus $\lambda_\$$ was defined by Materzok in [11]. We proved its correspondence with $\Lambda_\$$ in Section 3. In this section we proved that $\lambda$ can be reflected into $\Lambda_\$$. This fact can be weakened to equational correspondence. All of $\Lambda\mu_v$, $\lambda_\$$, $\Lambda_\$$ and $\lambda$ correspond equationally because equational correspondences are composable. □

## 5 Conclusion

We developed a novel calculus of delimited control $\Lambda_\$$ whose confluent reduction theory induces an axiomatisation that is both sound and complete with respect to denotational semantics provided by CPS translation. We have shown a close relationship (i.e. reflection) between reduction theories of pure lambda calculus and calculus $\Lambda_\$$, which justifies the assertion that `shift0`-style delimited control and call-by-name are diffent means of expressing the same computation.

Sabry and Felleisen [17] found an undirected axiomatisation of pure call-by-value lambda calculus and also of `call/cc`-style abortive control. Due to lack of interest in directedness, they amalgamated some rules of opposite direction into $\beta_\Omega$ axiom. Kameyama and Hasegawa [9] found an undirected axiomatisation of `shift`-style delimited control, while Materzok [11] axiomatised `shift0`. Our paper spells out the directed axiomatisation for the latter. Our rules are not only directed but also local (or fine-grained) which means that no recursive family of contexts is needed to express them.

Developments in this paper are untyped. We conjecture that it should be possible to adapt existing type systems for `shift0` [12,11,14]. We are advancing in the process of axiomatising algebraic effects with deep handlers, which are a related [8,14] approach to computational effects.

## References


[1] Biernacki, D., M. Pyzik and F. Sieczkowski, *A reflection on continuation-composing style*, in: Z. M. Ariola, editor, *5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference)*, volume 167 of *LIPIcs*, pages 18:1–18:17, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020).
https://doi.org/10.4230/LIPIcs.FSCD.2020.18

[2] Biernacki, D., M. Pyzik and F. Sieczkowski, *Reflecting stacked continuations in a fine-grained direct-style reduction theory*, in: N. Veltri, N. Benton and S. Ghilezan, editors, *PPDP 2021: 23rd International Symposium on Principles and Practice of Declarative Programming, Tallinn, Estonia, September 6-8, 2021*, pages 4:1–4:13, ACM (2021).
https://doi.org/10.1145/3479394.3479399

[3] Danvy, O. and A. Filinski, *Abstracting control*, in: G. Kahn, editor, *Proceedings of the 1990 ACM Conference on LISP and Functional Programming, LFP 1990, Nice, France, 27-29 June 1990*, pages 151–160, ACM (1990).
https://doi.org/10.1145/91556.91622

[4] Downen, P. and Z. M. Ariola, *Compositional semantics for composable continuations: from abortive to delimited control*, in: J. Jeuring and M. M. T. Chakravarty, editors, *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming, Gothenburg, Sweden, September 1-3, 2014*, pages 109–122, ACM (2014).
https://doi.org/10.1145/2628136.2628147

[5] Felleisen, M., *The theory and practice of first-class prompts*, in: J. Ferrante and P. Mager, editors, *Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, San Diego, California, USA, January 10-13, 1988*, pages 180–190, ACM Press (1988).
https://doi.org/10.1145/73560.73576

[6] Filinski, A., *Representing monads*, in: H. Boehm, B. Lang and D. M. Yellin, editors, *Conference Record of POPL'94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon, USA, January 17-21, 1994*, pages 446–457, ACM Press (1994).
https://doi.org/10.1145/174675.178047

[7] Filinski, A., *Representing layered monads*, in: A. W. Appel and A. Aiken, editors, *POPL '99, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999*, pages 175–188, ACM (1999).
https://doi.org/10.1145/292540.292557

[8] Forster, Y., O. Kammar, S. Lindley and M. Pretnar, *On the expressive power of user-defined effects: Effect handlers, monadic reflection, delimited control*, J. Funct. Program. **29**, page e15 (2019).
https://doi.org/10.1017/S0956796819000121

[9] Kameyama, Y. and M. Hasegawa, *A sound and complete axiomatization of delimited continuations*, in: C. Runciman and O. Shivers, editors, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, pages 177–188, ACM (2003).
https://doi.org/10.1145/944705.944722

[10] Kiselyov, O. and C. Shan, *A substructural type system for delimited continuations*, in: S. R. D. Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *Lecture Notes in Computer Science*, pages 223–239, Springer (2007).
https://doi.org/10.1007/978-3-540-73228-0_17

[11] Materzok, M., *Axiomatizing subtyped delimited continuations*, in: S. R. D. Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 521–539, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2013).
https://doi.org/10.4230/LIPIcs.CSL.2013.521

[12] Materzok, M. and D. Biernacki, *Subtyping delimited continuations*, in: M. M. T. Chakravarty, Z. Hu and O. Danvy, editors, *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming, ICFP 2011, Tokyo, Japan, September 19-21, 2011*, pages 81–93, ACM (2011).
https://doi.org/10.1145/2034773.2034786

[13] Materzok, M. and D. Biernacki, *A dynamic interpretation of the CPS hierarchy*, in: R. Jhala and A. Igarashi, editors, *Programming Languages and Systems - 10th Asian Symposium, APLAS 2012, Kyoto, Japan, December 11-13, 2012. Proceedings*, volume 7705 of *Lecture Notes in Computer Science*, pages 296–311, Springer (2012).
https://doi.org/10.1007/978-3-642-35182-2_21

[14] Piróg, M., P. Polesiuk and F. Sieczkowski, *Typed equivalence of effect handlers and delimited control*, in: H. Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, volume 131 of *LIPIcs*, pages 30:1–30:16, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019).
https://doi.org/10.4230/LIPIcs.FSCD.2019.30

[15] Plotkin, G. D. and M. Pretnar, *Handlers of algebraic effects*, in: G. Castagna, editor, *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5502 of *Lecture Notes in Computer Science*, pages 80–94, Springer (2009).
https://doi.org/10.1007/978-3-642-00590-9_7

[16] Pretnar, M., *An introduction to algebraic effects and handlers. Invited tutorial paper*, in: D. R. Ghica, editor, *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22-25, 2015*, volume 319 of *Electronic Notes in Theoretical Computer Science*, pages 19–35, Elsevier (2015).
https://doi.org/10.1016/j.entcs.2015.12.003

[17] Sabry, A. and M. Felleisen, *Reasoning about programs in continuation-passing style*, LISP Symb. Comput. **6**, pages 289–360 (1993).
https://doi.org/10.1007/BF01019462

[18] Sabry, A. and P. Wadler, *A reflection on call-by-value*, ACM Trans. Program. Lang. Syst. **19**, pages 916–941 (1997).
https://doi.org/10.1145/267959.269968

[19] Shan, C., *A static simulation of dynamic delimited control*, Higher-Order and Symbolic Computation **20**, pages 371–401 (2007).
https://doi.org/10.1007/s10990-007-9010-4


## A Extra proofs

**Lemma A.1 (3.5)** *Equality* $(M\,N)^* = \lambda\,k\,.\,M^*\,(\lambda\,x\,.\,N^*\,(\lambda\,y\,.\,x\,y\,k))$ *holds in* $\lambda$.

**Proof.** By cases (value/nonvalue) on $M$ and $N$.

- Case $M \equiv V$, $N \equiv W$.

$$\begin{aligned}(V\,W)^* &\equiv V^\dagger\,W^\dagger =_\eta \lambda\,k\,.\,V^\dagger\,W^\dagger\,k =_\beta \lambda\,k\,.(\lambda\,x\,.\,x\,W^\dagger\,k)\,V^\dagger =_\beta \lambda\,k\,.\,V^*\,(\lambda\,x\,.\,x\,W^\dagger\,k)\\ &=_\beta \lambda\,k\,.\,V^*\,(\lambda\,x\,.(\lambda\,y\,.\,x\,y\,k)\,W^\dagger) =_\beta \lambda\,k\,.\,V^*\,(\lambda\,x\,.\,W^*\,(\lambda\,y\,.\,x\,y\,k))\end{aligned}$$

- Case $M \equiv V$, $N \equiv Q$.

$$(V\,Q)^* \equiv \lambda\,k\,.\,Q^*\,(\lambda\,y\,.\,V^\dagger\,y\,k) =_\beta \lambda\,k\,.(\lambda\,x\,.\,Q^*\,(\lambda\,y\,.\,x\,y\,k))\,V^\dagger =_\beta \lambda\,k\,.\,V^*\,(\lambda\,x\,.\,Q^*\,(\lambda\,y\,.\,x\,y\,k))$$

- Case $M \equiv P$, $N \equiv W$.

$$(P\,W)^* \equiv \lambda\,k\,.\,P^*\,(\lambda\,x\,.\,x\,W^\dagger\,k) =_\beta \lambda\,k\,.\,P^*\,(\lambda\,x\,.(\lambda\,y\,.\,x\,y\,k)\,W^\dagger) =_\beta \lambda\,k\,.\,P^*\,(\lambda\,x\,.\,W^*\,(\lambda\,y\,.\,x\,y\,k))$$

- Case $M \equiv P$, $N \equiv Q$.

$$(P\,Q)^* \equiv \lambda\,k\,.\,P^*\,(\lambda\,x\,.(x\,Q)^*\,k) \equiv \lambda\,k\,.\,P^*\,(\lambda\,x\,.(\lambda\,k\,.\,Q^*\,(\lambda\,y\,.\,x\,y\,k))\,k) =_\beta \lambda\,k\,.\,P^*\,(\lambda\,x\,.\,Q^*\,(\lambda\,y\,.\,x\,y\,k))$$

□

**Lemma A.2 (4.11)** *$M^\#[N^\natural/x] \twoheadrightarrow_{\Lambda_\$} M[N/x]^\#$ and $M^\natural[N^\natural/x] \twoheadrightarrow_{\Lambda_\$} M[N/x]^\natural$ hold.*

**Proof.** By structural induction on $M$. We prove the first conjunct below.

- Case $M \equiv x$. $x^\#[N^\natural/x] \equiv \mathcal{S}_0(x)[N^\natural/x] \equiv \mathcal{S}_0(N^\natural) \rightarrow^?_{\text{Lemma 4.9}} N^\# \equiv x[N/x]^\#$.
- Case $M \equiv y$. $y^\#[N^\natural/x] \equiv \mathcal{S}_0(y)[N^\natural/x] \equiv \mathcal{S}_0(y) \equiv y^\# \equiv y[N/x]^\#$.
- Case $M \equiv M_1\,M_2$.

$$(M_1\,M_2)^\#[N^\natural/x] \equiv M_1^\natural[N^\natural/x]\,M_2^\natural[N^\natural/x] \twoheadrightarrow_{\text{IH}} M_1[N/x]^\natural\,M_2[N/x]^\natural \equiv (M_1\,M_2)[N/x]^\#$$

- Case $M \equiv \lambda\,y\,.\,M_1$. If $M_1 \equiv y\,M_2$ and $y$ is fresh wrt. $M_2$, then

$$(\lambda\,y\,.\,y\,M_2)^\#[N^\natural/x] \equiv M_2^\natural[N^\natural/x] \twoheadrightarrow_{\text{IH}} M_2[N/x]^\natural \equiv (\lambda\,y\,.\,y\,M_2)[N/x]^\#.$$

  Otherwise, $(\lambda\,y\,.\,M_1)^\#[N^\natural/x] \equiv \mathcal{S}_0\,y\,.\,M_1^\#[N^\natural/x] \twoheadrightarrow_{\text{IH}} \mathcal{S}_0\,y\,.\,M_1[N/x]^\natural \rightarrow^?_{\text{Lemma 4.9}} (\lambda\,y\,.\,M_1[N/x])^\# \equiv (\lambda\,y\,.\,M_1)[N/x]^\#$.

We now prove the second conjunct.

- Case $M \equiv x$. $x^\natural[N^\natural/x] \equiv x[N^\natural/x] \equiv N^\natural \equiv x[N/x]^\natural$.
- Case $M \equiv y$. $y^\natural[N^\natural/x] \equiv y[N^\natural/x] \equiv y \equiv y^\natural \equiv y[N/x]^\natural$.
- Case $M \equiv M_1\,M_2$.

$$(M_1\,M_2)^\natural[N^\natural/x] \equiv \$(M_1^\natural[N^\natural/x]\,M_2^\natural[N^\natural/x]) \twoheadrightarrow_{\text{IH}} \$(M_1[N/x]^\natural\,M_2[N/x]^\natural) \equiv (M_1\,M_2)[N/x]^\natural$$

- Case $M \equiv \lambda\,y\,.\,M_1$. $(\lambda\,y\,.\,M_1)^\natural[N^\natural/x] \equiv \lambda\,y\,.\,M_1^\#[N^\natural/x] \twoheadrightarrow_{\text{IH}} \lambda\,y\,.\,M_1[N/x]^\# \equiv (\lambda\,y\,.\,M_1)[N/x]^\natural$.

□