Under consideration for publication in J. Functional Programming

Call-by-name extensionality and confluence

PHILIP JOHNSON-FREYD, PAUL DOWNEN and ZENA M. ARIOLA
University of Oregon, Eugene Oregon, USA
(*e-mail:* {philipjf, pdownen, ariola}@cs.uoregon.edu)

Abstract

Designing rewriting systems that respect functional extensionality for call-by-name languages with effects turns out to be surprisingly challenging. Simply interpreting extensional laws like η as reduction rules easily breaks confluence. We explore these issues in the setting of a sequent calculus. Building on an insight that appears in different aspects of the theory of call-by-name functional languages—confluent rewriting for two independent control calculi and sound continuation-passing style transformations—we give a confluent reduction system for lazy extensional functions. Finally, we consider limitations to this approach when used for strict evaluation and types beyond functions.

1 Introduction

Extensionality helps make syntactic theories less syntactic—objects that behave the same are the same even though they appear to be different. However, combining a call-by-name parameter passing technique with extensionality presents unexpected challenges. For example, a useful property for rewriting theories is confluence, which says that the chosen order of reductions does not impact the result—all diverging paths will eventually meet back up in the end. In a logic, confluence guarantees consistency so that we do not end up equating `true` to `false`. It also provides a good vehicle to conduct proofs establishing a standard reduction path, thus demonstrating that the rewriting theory can be implemented in a deterministic fashion. Indeed, the λ-calculus with the extensional η law is confluent (Barendregt, 1984). However, once the calculus is extended with control effects, confluence is lost (David & Py, 2001). Even worse, just adding extensional products with a surjectivity law breaks confluence of the pure λ-calculus as well (Klop & de Vrijer, 1989).

Given these troublesome problems, we might be inclined to conclude that confluence is too syntactic and low-level. After all, the consistency of a logic can be demonstrated using alternative techniques (Klop & de Vrijer, 1989), and if we are only interested in capturing program execution, we can focus directly on an operational semantics instead of starting with such a flexible rewriting system. These are sensible arguments, and indeed there are cases where more abstract notions of confluence are needed, as in a calculus with cyclic structures (Ariola & Blom, 2002). However, it might come as a surprise that extensionality creates issues even when focusing on the pure λ-calculus alone without any extensions. For example, continuation-passing style transformations hard-wire the evaluation strategy of a language directly in the program itself. Naturally, we expect that equivalent source programs continue to be equivalent after the transformation. But under a known call-by-name continuation-passing style transformation due to Plotkin (Plotkin, 1975), β-equivalence is preserved while η-equivalence is violated.

Since these issues surrounding extensionality show up in multiple disconnected places, we wonder if they act as the canary in a coal mine, signaling that the pieces of the calculus do not quite fit perfectly together and that we should question our basic assumptions. Indeed, both the efforts to solve the confluence problem and the search for a sound continuation-passing style transformation have made use of the same insight about the nature of lazy functions. The goal of this paper is to present this alternative and fundamental view of lazy functions in an incremental way, in a setting that brings out better intuition about the insight. We do not claim originality of the solution, however, we hope the framework used here will serve as a more suitable starting point for tackling the remaining issues of integrating extensionality with other structures.

The central idea developed in later articles, and which in hindsight can be recognized in at least Danos, Joinet & Schellinx, 1997, is that functions are co-data (Zeilberger, 2009; Downen & Ariola, 2014). That is, they are not concretely constructed objects like tuples in an ML-like language, but rather abstract objects that only respond to messages. The co-data nature of functions suggests a shift in focus away from functions themselves and toward the *observations* that use them. How can we use a function? We need to provide an argument and a return point that does something with the result. This highlights the fact that the observations on functions are constructed, and we refer to this structure as a *call stack* (Pitts, 2000). Thus, a function is a destructor for call stacks. Concretely, that means that we want to treat “inside out” contexts as primitive (Felleisen, Wand, Friedman & Duba, 1988). Since we want both functions and their observations to have equal status, we leave the λ-calculus and instead embrace the sequent calculus as a language that clearly delineates those who produce from those who consume.

Another reason for using the sequent calculus is that control is inherent in the framework rather than being added as an afterthought. But what does control have to do with functions? The two seem unrelated because we are accustomed to associating control with operators like Scheme’s `call/cc`, and indeed if we assume that control is just for expressing `call/cc` then there is little justification for using it to explain functions. Instead, control should be seen as a way to give a name to the call stack, similar to the way we name values (Curien & Munch-Maccagnoni, 2010). Thus, much as we match on the structure of a tuple, functions are objects that match on the structure of a call stack. Note that this view of functions as call-stack destructors is independent of the evaluation strategy. It is not tied to call-by-name but is inherent to the primordial notion of a function itself, so it holds just as well in a call-by-value setting (Zeilberger, 2009).

All three of these ideas—functions as pattern matching, contexts as primitive, and control as the naming of contexts—have recently received an extensive analysis and justification from a logical perspective (Munch-Maccagnoni & Scherer, 2015).

What, then, is the essence of a lazy function? The essential ingredient is that lazy functions perform a lazy pattern match on their call stack, so functional extensionality is captured by the surjectivity of the call stack as a pairing construct. Indeed, this same insight has sprung up as the key linchpin to the problems surrounding the implementation of the η law. An alternative call-by-name continuation-passing translation based on pairs arises from a polarized translation of classical logic (Lafont, Reus, & Streicher, 1993; Girard, 1991). Moreover, at least as long as the pairs in the target are categorical products, that is, surjective pairs, this translation validates the η law (Hofmann & Streicher, 2002; Fujita, 2003). Surjective call stacks have also been used to establish an extensional confluent rewriting system in two independently developed calculi: the $\Lambda\mu$-calculus (Nakazawa & Nagai, 2014), which is interesting for its connections with Böhm separability (Saurin, 2010) and delimited control (Herbelin & Ghilezan, 2008); and the stack calculus, which can be seen as the *dual* to natural deduction (Carraro, Ehrhard and Salibra, 2012). It is surprising to see that these different problem domains unveil the same key insight about lazy functions.

In the end, we have a satisfying solution to confluence in the λ-calculus with both extensionality and classical control effects as well as a sound call-by-name continuation-passing style transformation. This immediately invites the question: does the solution scale up to more realistic calculi with other structures, like products and sums, found in programming languages? Unfortunately, the answer is a frustrating "no." The solution does not easily apply to programming constructs like sum types, which involve decision points with multiple branches. Worse, confluence appears to break down again when we combine both data and co-data. However, since the use of surjective call stacks works so well for functions in isolation, we see these roadblocks as prompts for a new avenue of study into extensional rewriting systems for other programming structures.

1.1 Contents

We begin our exploration of extensionality and control with an extensional call-by-name sequent calculus that has a traditional presentation of functions as λ-abstraction (Figure 1), and contrast this traditional view with an alternative one based on pattern matching (Figure 3), which we show is equivalent to the first (Section 2). Next, we show how the naïve rewriting theory for this sequent calculus lacks confluence and present an alternative rewriting theory that rewrites pattern matching into projections (Section 3). The projection-based system is presented in both sequent (Figure 5) and natural deduction style (Figure 7), and the two styles are shown to be in equational correspondence. This new projection-based formulation of functions is then shown to be equivalent to the two previous presentations (Section 4). Confluence of the extensional rewriting theory is shown by reducing the full calculus down to a small core calculus (Figure 12) which is known to be confluent (Section 5). Finally, we discuss how this solution to confluence can break down when the calculus is extended with other structures from programming languages (Section 6).

2 An extensional call-by-name equational theory

As a native language for studying first-class control we consider $\bar{\lambda}\mu\tilde{\mu}_\eta$ (Figure 1), an extensional version of a call-by-name sequent calculus (Curien & Herbelin, 2000). Note that throughout this paper we present rules with the implicit assumption that they respect static scope: rules are always restricted to avoid variable capture or escape. Although our focus is on the underlying untyped term language, $\bar{\lambda}\mu\tilde{\mu}_\eta$ is motivated by its typed version (Figure 2) which corresponds to a two-sided classical sequent calculus in much the same way that the simply typed λ-calculus corresponds to intuitionistic natural deduction.

$$
\begin{aligned}
c \in \text{Command} &::= \langle v \| e \rangle \\
v \in \text{Terms} &::= x \mid \lambda x.v \mid \mu\alpha.c \\
e \in \text{Co-Terms} &::= \alpha \mid v \cdot e \mid \tilde{\mu}x.c \\
E \in \text{Co-Values} &::= \alpha \mid v \cdot E
\end{aligned}
$$

$$
\begin{aligned}
\langle \mu\alpha.c \| E \rangle &=_{\mu} c[E/\alpha] \\
\langle v \| \tilde{\mu}x.c \rangle &=_{\tilde{\mu}} c[v/x] \\
\mu\alpha.\langle v \| \alpha \rangle &=_{\eta_\mu} v \\
\tilde{\mu}x.\langle x \| e \rangle &=_{\eta_{\tilde{\mu}}} e \\
\langle \lambda x.v' \| v \cdot e \rangle &=_{\beta} \langle v \| \tilde{\mu}x.\langle v' \| e \rangle \rangle \\
\lambda x.\mu\alpha.\langle v \| x \cdot \alpha \rangle &=_{\eta} v
\end{aligned}
$$

Fig. 1. The call-by-name $\bar{\lambda}\mu\tilde{\mu}$-calculus extended with η ($\bar{\lambda}\mu\tilde{\mu}_\eta$)

$$
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A \mid \Delta}[id_R] \qquad \frac{(\alpha : A) \in \Delta}{\Gamma \mid \alpha : A \vdash \Delta}[id_L]
$$

$$
\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}[Act_R] \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}[Act_L]
$$

$$
\frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta}{\langle v \| e \rangle : (\Gamma \vdash \Delta)}[cut]
$$

$$
\frac{\Gamma, x : A \vdash v : B \mid \Delta}{\Gamma \vdash \lambda x.v : A \to B \mid \Delta}[\to_R] \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid v \cdot e : A \to B \vdash \Delta}[\to_L]
$$

Fig. 2. Simple type assignment for $\bar{\lambda}\mu\tilde{\mu}_\eta$

Since $\bar{\lambda}\mu\tilde{\mu}_\eta$ is based on the sequent calculus, where proof dynamics is captured by the elimination of cuts, programs are written as two-sided entities called *commands*, where the two sides represent two opposing forces of computation—the production (via *terms*) and consumption (via *co-terms*) of information—corresponding to the two sides of a cut. For example, a λ-abstraction $\lambda x.v$, as is written in the λ-calculus, is a term that describes the necessary behavior for creating a functional value, whereas a call stack $v \cdot e$ is a co-term that describes the method for using a function. Additionally, we have generic abstractions which allow each side to give a name to the other, regardless of their specific form. The $\tilde{\mu}$-abstraction $\tilde{\mu}x.c$ is a consumer which names its input x before running the command c. Dually, the μ-abstraction $\mu\alpha.c$ is a producer which names the location for its output α before running c. The typing judgment for terms corresponds to the collection of proofs with an *active* formula on the right, and dually, co-terms correspond to proofs with an active formula on the left. Readers unaccustomed to the sequent calculus may find it helpful to think of co-terms as contexts in the lambda calculus and the construct $\langle v \| e \rangle$ as filling the hole in e with the term v. In this view, $\tilde{\mu}x.c$ corresponds roughly to `let` $x = \square$ `in` c while $v \cdot e$ corresponds to the context $e[\square\ v]$. We can see that $\tilde{\mu}$ allows us to build more complicated compound contexts, beyond just the applicative contexts formed with the call stack constructor (*i.e.,* the $\cdot$ constructor). Similarly, μ-abstraction lets us build more complicated compound terms beyond just λ. Indeed, the μ operator can be seen as arising as a solution to the problem of how to encode elimination forms, and their associated reduction equations, into an abstract machine language like $\bar{\lambda}\mu\tilde{\mu}_\eta$ (Curien & Munch-Maccagnoni, 2010; Munch-Maccagnoni & Scherer, 2015). In the Krivine abstract machine (Krivine, 2007), the function application construct from the lambda calculus is handled by way of a reduction rule

$$\langle v\ v' \| E\rangle \to \langle v \| v' \cdot E\rangle$$

which refocuses (Danvy & Nielsen, 2004) the command to incrementally replace a outside-in context with its associated inside-out co-value. Indeed, we can take this reduction rule as defining the meaning of application as a syntactic construct for extending the context. However, in order to directly define application internally to the calculus we need a way to define terms by giving a name to their context. Thus, $v\ v'$ can be represented as the compound term $\mu\alpha.\langle v \| v' \cdot \alpha\rangle$ which implements the same reduction rule.

The μ-abstraction $\mu\alpha.c$ can also be read as `call/cc`$(\lambda\alpha.c)$, and a co-variable α is akin to a context of the form `throw` $\alpha\ \Box$, which invokes a stored continuation. However, we find the sequent calculus presentation instructive as it emphasizes symmetries, reducing the number of distinct concepts. In a λ-calculus presentation, control operators often come across as exotic and are difficult to build an intuition about. Here, μ-abstraction is simply understood as providing a name to an output channel in a computation: it is no more complicated than $\tilde{\mu}$-abstraction which provides a name to an input.

$$\begin{aligned}
c \in \text{Command} &::= \langle v \| e\rangle \\
v \in \text{Terms} &::= x \mid \mu[(x\cdot\alpha).c] \mid \mu\alpha.c \\
e \in \text{Co-Terms} &::= \alpha \mid v\cdot e \mid \tilde{\mu}x.c \\
E \in \text{Co-Values} &::= \alpha \mid v\cdot E
\end{aligned}$$

$$\begin{aligned}
\langle \mu\alpha.c \| E\rangle &=_\mu c[E/\alpha] \\
\langle v \| \tilde{\mu}x.c\rangle &=_{\tilde{\mu}} c[v/x] \\
\mu\alpha.\langle v \| \alpha\rangle &=_{\eta_\mu} v \\
\tilde{\mu}x.\langle x \| e\rangle &=_{\eta_{\tilde{\mu}}} e \\
\langle \mu[(x\cdot\alpha).c] \| v\cdot e\rangle &=_\beta \langle v \| \tilde{\mu}x.\langle \mu\alpha.c \| e\rangle\rangle \\
\mu[(x\cdot\alpha).\langle v \| x\cdot\alpha\rangle] &=_\eta v
\end{aligned}$$

Fig. 3. The call-by-name $\mu\tilde{\mu}_\eta^{\rightarrow}$-calculus

As a contrast to λ-abstractions, we adopt an alternative notation for functions (Curien & Munch-Maccagnoni, 2010), shown in Figure 3. A function is not constructed; instead it is a procedure that reacts by pattern matching on its observing call stack. As such, we write a function as $\mu[(x\cdot\alpha).c]$, which says that the context is decomposed into an argument named x and a place named α for the result. Indeed, either representation for functions can be seen as syntactic sugar for the other:

$$\mu[(x\cdot\alpha).c] \triangleq \lambda x.\mu\alpha.c \qquad \text{or} \qquad \lambda x.v \triangleq \mu[(x\cdot\alpha).\langle v \| \alpha\rangle] \tag{1}$$

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A\mid\Delta}[id_R] \qquad \frac{(\alpha:A)\in\Delta}{\Gamma\mid\alpha:A\vdash\Delta}[id_L]$$

$$\frac{c:(\Gamma\vdash\alpha:A,\Delta)}{\Gamma\vdash\mu\alpha.c:A\mid\Delta}[Act_R] \qquad \frac{c:(\Gamma,x:A\vdash\Delta)}{\Gamma\mid\tilde{\mu}x.c:A\vdash\Delta}[Act_L]$$

$$\frac{\Gamma\vdash v:A\mid\Delta \qquad \Gamma\mid e:A\vdash\Delta}{\langle v\|e\rangle:(\Gamma\vdash\Delta)}[cut]$$

$$\frac{c:(\Gamma,x:A\vdash\alpha:B,\Delta)}{\Gamma\vdash\mu[(x\cdot\alpha).c]:A\to B\mid\Delta}[\to_R] \qquad \frac{\Gamma\vdash v:A\mid\Delta \qquad \Gamma\mid e:B\vdash\Delta}{\Gamma\mid v\cdot e:A\to B\vdash\Delta}[\to_L]$$

Fig. 4. Simple type assignment for $\mu\tilde{\mu}_\eta^{\to}$

From a logical standpoint, this alternative interpretation of functions corresponds to the right introduction rule for implication, which does not preserve activation on the right. This can be seen in the type assignment for the $\mu\tilde{\mu}_\eta^{\to}$-calculus (Figure 4).

In order to make the equivalence between these two different views of functions more precise, we utilize the concept of an equational correspondence (Sabry & Felleisen, 1993). To distinguish the different equational theories, we use the notation $T\vdash t=t'$ to indicate that t and t' are equated by the theory T. Given two equational theories S and T, the translations $\flat : S\to T$ and $\sharp : T\to S$ form an *equational correspondence* whenever the following four conditions hold:

1. ($\flat$) For all terms s_1 and s_2 of S, $S\vdash s_1=s_2$ implies $T\vdash s_1^\flat=s_2^\flat$.
2. ($\sharp$) For all terms t_1 and t_2 of T, $T\vdash t_1=t_2$ implies $S\vdash t_1^\sharp=t_2^\sharp$.
3. ($\flat\sharp$) For all terms s of S, $S\vdash s=(s^\flat)^\sharp$.
4. ($\sharp\flat$) For all terms t of T, $T\vdash t=(t^\sharp)^\flat$.

Proposition 1
$\bar{\lambda}\mu\tilde{\mu}_\eta$ and $\mu\tilde{\mu}_\eta^{\to}$ are in equational correspondence.

Proof
The translations of the equational correspondence are given by the macro definitions between the two syntactic representations of functions (Equation 1). The translation is not a syntactic isomorphism, because $\lambda x.v$ in $\bar{\lambda}\mu\tilde{\mu}_\eta$ becomes $\mu[(x\cdot\alpha).\langle v\|\alpha\rangle]$ in $\mu\tilde{\mu}_\eta^{\to}$, which in turn translates back to $\bar{\lambda}\mu\tilde{\mu}_\eta$ as $\lambda x.\mu\alpha.\langle v\|\alpha\rangle$. However, we have

$$\lambda x.v =_{\eta_\mu} \lambda x.\mu\alpha.\langle v\|\alpha\rangle$$

in $\bar{\lambda}\mu\tilde{\mu}_\eta$. Similarly, $\mu[(x\cdot\alpha).c]$ in $\mu\tilde{\mu}_\eta^{\to}$ becomes $\lambda x.\mu\alpha.c$, which roundtrips back to the term $\mu[(x\cdot\beta).\langle\mu\alpha.c\|\beta\rangle]$, but

$$\mu[(x\cdot\alpha).c] =_\alpha \mu[(x\cdot\beta).c[\beta/\alpha]] =_\mu \mu[(x\cdot\beta).\langle\mu\alpha.c\|\beta\rangle] \ .$$

All other syntactic constructs roundtrip to themselves exactly. The equational correspondence is completed by observing that all axioms of $\bar{\lambda}\mu\tilde{\mu}_\eta$ are derivable in $\mu\tilde{\mu}_\eta^{\to}$ after translation, and vice versa. ■

3 An extensional call-by-name reduction theory

Having seen the equational theory for the $\mu\tilde{\mu}_{\eta}^{\rightarrow}$-calculus, we now look for a corresponding confluent reduction theory. The simplest starting point is to take the left-to-right reading of each axiom as a reduction. However, this system lacks confluence due to a conflict between the η and μ rules:

$$\mu\delta.\langle y\|\beta\rangle \leftarrow_{\eta} \mu[(x\cdot\alpha).\langle\mu\delta.\langle y\|\beta\rangle\|x\cdot\alpha\rangle] \rightarrow_{\mu} \mu[(x\cdot\alpha).\langle y\|\beta\rangle] \tag{2}$$

This issue is not caused by the two-sided sequent calculus presentation; it is part of a fundamental conflict between lazy functions and control. Indeed, a similar issue occurs in the $\lambda\mu$-calculus (Parigot, 1992), a language with control based on the λ-calculus (David & Py, 2001); given a term of the form $\lambda x.M\ x$, a reduction in $M\ x$ could ruin the pattern for the η rule. This phenomenon does not occur in plain λ-calculus since both reducts are the same. To combat this issue, David and Py introduce a new rule, written in $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ as

$$\mu\delta.c \rightarrow_{\nu} \mu[(x\cdot\alpha).c[x\cdot\alpha/\delta]]$$

The above diverging diagram can thus be brought back together:

$$\mu\delta.\langle y\|\beta\rangle \rightarrow_{\nu} \mu[(x\cdot\alpha).\langle y\|\beta\rangle]$$

The ν rule can be understood as performing not an η-*reduction* but rather an η-*expansion* followed by a μ-reduction.

$$\mu\delta.c \leftarrow_{\eta} \mu[(x\cdot\alpha).\langle\mu\delta.c\|x\cdot\alpha\rangle] \rightarrow_{\mu} \mu[(x\cdot\alpha).c[x\cdot\alpha/\delta]]$$

Because ν involves expansion, it risks eliminating the concept of (finite) normal forms. Moreover, confluence is not restored for arbitrary open terms, since terms with a free co-variable can still lack confluence:

$$\langle y\|\beta\rangle \leftarrow_{\mu} \langle\mu\delta.\langle y\|\beta\rangle\|\delta\rangle \leftarrow_{\eta} \langle\mu[(x\cdot\alpha).\langle\mu\delta.\langle y\|\beta\rangle\|x\cdot\alpha\rangle]\|\delta\rangle \rightarrow_{\mu} \langle\mu[(x\cdot\alpha).\langle y\|\beta\rangle]\|\delta\rangle$$

The $\Lambda\mu_{\text{cons}}$-calculus (Nakazawa & Nagai, 2014) provides an interesting alternative solution to the conflict between functional extensionality and control. The $\Lambda\mu_{\text{cons}}$-calculus is an extension of the $\lambda\mu$-calculus (Parigot, 1992) which adds not only an explicit representation of call stacks as co-terms, similar to $\mu\tilde{\mu}_{\eta}^{\rightarrow}$, but also projections *out* of these call stacks: `car` projects out the argument and `cdr` projects out the return continuation. This calculus suggests a third interpretation of functions based on projections, instead of either λ-abstractions or pattern matching. We can transport this alternative interpretation of functions to the sequent calculus by extending $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ with the new term $\texttt{car}(e)$, co-term $\texttt{cdr}(e)$, and co-value $\texttt{cdr}(E)$, giving us the $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ reduction theory in Figure 5. Note that to avoid spurious infinite reduction sequences, the ς-rules are restricted to only lift out non-co-values. Effectively, the `exp` rule implements functions as a μ-abstraction with projections out of the given call stack. Thus, although $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ does not have a direct reduction rule corresponding to the β-law, function calls are still derivable:

$$\begin{aligned}
\langle\mu[(x\cdot\alpha).c]\|v\cdot E\rangle &\rightarrow_{\texttt{exp}} \langle\mu\beta.c[\texttt{car}(\beta)/x,\texttt{cdr}(\beta)/\alpha]\|v\cdot E\rangle\\
&\rightarrow_{\mu} c[\texttt{car}(v\cdot E)/x,\texttt{cdr}(v\cdot E)/\alpha]\\
&\rightarrow_{\texttt{cdr}}\rightarrow_{\texttt{car}} c[v/x,E/\alpha]
\end{aligned}$$

$$\begin{aligned}
c \in \text{Command} &::= \langle v \| e \rangle \\
v \in \text{Terms} &::= x \mid \mu\alpha.c \mid \mu[(x\cdot\alpha).c] \mid \texttt{car}(e) \\
e \in \text{Co-Terms} &::= \alpha \mid v \cdot e \mid \tilde{\mu}x.c \mid \texttt{cdr}(e) \\
E \in \text{Co-Values} &::= \alpha \mid v \cdot E \mid \texttt{cdr}(E)
\end{aligned}$$

$$\begin{aligned}
\langle \mu\alpha.c \| E \rangle &\to_{\mu} c[E/\alpha] \\
\langle v \| \tilde{\mu}x.c \rangle &\to_{\tilde{\mu}} c[v/x] \\
\mu\alpha.\langle v \| \alpha \rangle &\to_{\eta_\mu} v \\
\tilde{\mu}x.\langle x \| e \rangle &\to_{\eta_{\tilde{\mu}}} e \\
\texttt{car}(E)\cdot\texttt{cdr}(E) &\to_{\texttt{surj}} E \\
\texttt{car}(v\cdot E) &\to_{\texttt{car}} v \\
\texttt{cdr}(v\cdot E) &\to_{\texttt{cdr}} E \\
\texttt{car}(e) &\to_{\varsigma_{\texttt{car}}} \mu\alpha.\langle \mu\beta.\langle \texttt{car}(\beta) \| \alpha \rangle \| e \rangle && e \notin \text{Co-Values} \\
\texttt{cdr}(e) &\to_{\varsigma_{\texttt{cdr}}} \tilde{\mu}x.\langle \mu\alpha.\langle x \| \texttt{cdr}(\alpha) \rangle \| e \rangle && e \notin \text{Co-Values} \\
v\cdot e &\to_{\varsigma.} \tilde{\mu}x.\langle \mu\alpha.\langle x \| v\cdot\alpha \rangle \| e \rangle && e \notin \text{Co-Values} \\
\mu[(x\cdot\alpha).c] &\to_{\texttt{exp}} \mu\beta.c[\texttt{car}(\beta)/x, \texttt{cdr}(\beta)/\alpha]
\end{aligned}$$

Fig. 5. The $\mu\tilde{\mu}^{\to}_{\text{cons}}$ reduction theory

Moreover, even the η-law becomes derivable as a sequence of reductions:

$$\mu[(x\cdot\alpha).\langle v \| x\cdot\alpha \rangle] \to_{\texttt{exp}} \mu\beta.\langle v \| \texttt{car}(\beta)\cdot\texttt{cdr}(\beta) \rangle \to_{\texttt{surj}} \mu\beta.\langle v \| \beta \rangle \to_{\eta_\mu} v$$

Yet even though both μ and η are derivable in $\mu\tilde{\mu}^{\to}_{\text{cons}}$, the exp rule brings our previous critical pairs back together (as usual, $\twoheadrightarrow$ stands for multiple steps of reduction):

$$\langle y \| \beta \rangle \twoheadleftarrow \langle \mu[(x\cdot\alpha).\langle \mu\delta.\langle y \| \beta \rangle \| x\cdot\alpha \rangle] \| \delta \rangle \to_{\mu} \langle \mu[(x\cdot\alpha).\langle y \| \beta \rangle] \| \delta \rangle \to_{\texttt{exp}} \langle \mu\gamma.\langle y \| \beta \rangle \| \delta \rangle \to_{\mu} \langle y \| \beta \rangle$$

From a logical standpoint, the new forms $\texttt{car}(-)$ and $\texttt{cdr}(-)$ correspond to *elimination* rules for implication (Figure 6). Note that the $\texttt{cdr}(-)$ rule is a *left* elimination rule.

Before showing that the $\mu\tilde{\mu}^{\to}_{\text{cons}}$ reduction theory is indeed confluent we need to demonstrate that its associated equational theory—obtained as the symmetric, transitive and reflexive closure of its reductions—is equivalent to $\mu\tilde{\mu}^{\to}_{\eta}$.

$$\frac{\Gamma \mid e : A \to B \vdash \Delta}{\Gamma \vdash \texttt{car}(e) : A \mid \Delta}[\to \mathit{Elim}_1] \quad \frac{\Gamma \mid e : A \to B \vdash \Delta}{\Gamma \mid \texttt{cdr}(e) : B \vdash \Delta}[\to \mathit{Elim}_2]$$

Fig. 6. Additional typing rules for $\mu\tilde{\mu}^{\to}_{\text{cons}}$

3.1 Bridging sequent calculus and natural deduction

Since $\mu\tilde{\mu}^{\to}_{\text{cons}}$ is designed as a sequent calculus counterpart to the $\Lambda\mu_{\text{cons}}$-calculus, they are predictably related to one another. In particular, the $\mu\tilde{\mu}^{\to}_{\text{cons}}$-calculus is equivalent to the $\lambda\mu_{\text{cons}}$-calculus shown in Figure 7, which syntactically distinguishes between commands and terms, as did the original $\lambda\mu$-calculus (Parigot, 1992). The type assignment for $\lambda\mu_{\text{cons}}$ given in Figure 8 differs from the type system given in (Nakazawa & Nagai, 2014) due

$$\begin{aligned}
v \in \text{Terms} &::= x \mid \lambda x.v \mid v\,v \mid \mu\alpha.c \mid \texttt{car}\ S\\
S \in \text{Streams} &::= \alpha \mid v :: S \mid \texttt{cdr}\ S\\
c \in \text{Commands} &::= [S]v
\end{aligned}$$

$$\begin{aligned}
(\mu\alpha.c)\ v &=_{\beta_T} \mu\alpha.c[v :: \alpha/\alpha]\\
[S](\mu\alpha.c) &=_{\mu} c[S/\alpha]\\
\lambda x.v &=_{\text{exp}} \mu\alpha.[\texttt{cdr}\ \alpha](v[(\texttt{car}\ \alpha)/x])\\
[v' :: S]v &=_{\text{assoc}} [S](v\ v')\\
\texttt{car}\ (v :: S) &=_{\text{car}} v\\
\texttt{cdr}\ (v :: S) &=_{\text{cdr}} S\\
\mu\alpha.[\alpha]v &=_{\eta_\mu} v\\
(\texttt{car}\ S) :: (\texttt{cdr}\ S) &=_{\eta_{::}} S\\
[\texttt{cdr}\ S](v\ (\texttt{car}\ S)) &=_{\eta'_{::}} [S]v
\end{aligned}$$

Fig. 7. The $\lambda\mu_{\text{cons}}$ calculus

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A \mid \Delta}[id_R] \qquad \frac{(\alpha : A) \in \Delta}{\Gamma \mid \alpha : A \vdash \Delta}[id_L]$$

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}[Act] \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid S : A \vdash \Delta}{[S]v : (\Gamma \vdash \Delta)}[cut]$$

$$\frac{\Gamma \vdash v : A \to B \mid \Delta \qquad \Gamma \vdash v' : A \mid \Delta}{\Gamma \vdash v\ v' : B \mid \Delta}[Modus\ Ponens]$$

$$\frac{\Gamma, x : A \vdash v : B \mid \Delta}{\Gamma \vdash \lambda x.v : A \to B \mid \Delta}[\to_R] \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid S : B \vdash \Delta}{\Gamma \mid v :: S : A \to B \vdash \Delta}[\to_L]$$

$$\frac{\Gamma \mid S : A \to B \vdash \Delta}{\Gamma \vdash \texttt{car}\ S : A \mid \Delta}[\to Elim_1] \qquad \frac{\Gamma \mid S : A \to B \vdash \Delta}{\Gamma \mid \texttt{cdr}\ S : B \vdash \Delta}[\to Elim_2]$$

Fig. 8. Simple Type Assignment for $\lambda\mu_{\text{cons}}$

to this syntactic difference, making it closer to a traditional logic. In particular, $\lambda\mu_{\text{cons}}$ is typed according to the rules of classical natural deduction extended with left introduction and elimination rules. We can translate between $\mu\tilde{\mu}^{\to}_{\text{cons}}$ and $\lambda\mu_{\text{cons}}$ based on the standard relationship between the sequent calculus and $\lambda\mu$-calculus (Curien & Herbelin, 2000), as shown in Figure 9. The primary complication in translating from $\mu\tilde{\mu}^{\to}_{\text{cons}}$ to $\lambda\mu_{\text{cons}}$ is the usual issue that comes up when comparing sequent-based and λ-based languages: the sequent calculus language has additional syntactic categories that must be merged together in a λ-calculus language. In particular, the commands and terms of the two calculi are in correspondence, as are the co-values of $\mu\tilde{\mu}^{\to}_{\text{cons}}$ and streams of $\lambda\mu_{\text{cons}}$ (as shown by the auxiliary translation E^-). However, even though co-values and streams correspond, the syntactic treatment of general co-terms in $\mu\tilde{\mu}^{\to}_{\text{cons}}$ is absent in $\lambda\mu_{\text{cons}}$. For example, $\texttt{cdr}(\tilde{\mu}x.c)$ cannot be represented directly as a stream in $\lambda\mu_{\text{cons}}$.

$$(-)^\sharp : \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \to \lambda\mu_{\text{cons}}$$

$$\langle v \| e \rangle^\sharp \triangleq e^\sharp[v^\sharp]$$

$$\begin{array}{rlrl}
x^\sharp & \triangleq x & \alpha^\sharp & \triangleq [\alpha]\Box \\
(\mu\alpha.c)^\sharp & \triangleq \mu\alpha.(c^\sharp) & (\tilde{\mu}x.c)^\sharp & \triangleq [\delta]((\lambda x.\mu\delta.(c^\sharp))\Box) \\
\mu[(x\cdot\alpha).c]^\sharp & \triangleq \lambda x.\mu\alpha.(c^\sharp) & (v\cdot e)^\sharp & \triangleq e^\sharp[\Box\ v^\sharp] \\
\texttt{car}(e)^\sharp & \triangleq \mu\alpha.e^\sharp[\mu\beta.[\alpha](\texttt{car}\ \beta)] & \texttt{cdr}(e)^\sharp & \triangleq e^\sharp[\mu\alpha.[\texttt{cdr}\ \alpha]\Box]
\end{array}$$

$$(-)^- : \text{Co-Values} \to \text{Streams}$$

$$\alpha^- \triangleq \alpha \qquad (v\cdot E)^- \triangleq v^\sharp :: E^- \qquad \texttt{cdr}(E)^- \triangleq \texttt{cdr}(E^-)$$

$$(-)^\flat : \lambda\mu_{\text{cons}} \to \mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$$

$$\begin{array}{rlrlrl}
([S]v)^\flat & \triangleq \langle v^\flat \| S^\flat \rangle & (\lambda x.v)^\flat & \triangleq \mu[(x\cdot\alpha).\langle v^\flat \| \alpha \rangle] & \alpha^\flat & \triangleq \alpha \\
x^\flat & \triangleq x & (v_1\ v_2)^\flat & \triangleq \mu\alpha.\langle v_1^\flat \| v_2^\flat \cdot \alpha \rangle & (v :: S)^\flat & \triangleq v^\flat \cdot S^\flat \\
(\mu\alpha.c)^\flat & \triangleq \mu\alpha.c^\flat & (\texttt{car}\ S)^\flat & \triangleq \texttt{car}(S^\flat) & (\texttt{cdr}\ S)^\flat & \triangleq \texttt{cdr}(S^\flat)
\end{array}$$

Fig. 9. Translations from $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ to $\lambda\mu_{\text{cons}}$ and vice versa

To help bridge the gap between the two calculi, we introduce in $\lambda\mu_{\text{cons}}$ the notion of a *context*, denoted by the metavariable C, that is simply a command with a term-shaped hole in it. These contexts in $\lambda\mu_{\text{cons}}$ correspond to co-terms in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$. For example, the $\sharp$-translation of the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ call stack $x\cdot\alpha$ becomes the context $[\alpha](\Box\ x)$ in the $\lambda\mu_{\text{cons}}$-calculus, as opposed to the more direct translation as a stream $(x\cdot\alpha)^- = x :: \alpha$. Given any such context C, its translation as a $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-term is defined as a $\tilde{\mu}$-abstraction:

$$C^\flat \triangleq \tilde{\mu}x.(C[x])^\flat$$

With this additional technical detail for dealing with contexts due to the loss of co-terms, we form an equational correspondence between $\lambda\mu_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$.

Theorem 1
$\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ and $\lambda\mu_{\text{cons}}$ are in equational correspondence.

Proof
The equational correspondence between $\lambda\mu_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ is between the four different syntactic categories of the two calculi according to the following translations:

1. $\lambda\mu_{\text{cons}}$ commands correspond to $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ commands by $c^\flat$ and $c^\sharp$,
2. $\lambda\mu_{\text{cons}}$ terms correspond to $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ terms by $v^\flat$ and $v^\sharp$,
3. $\lambda\mu_{\text{cons}}$ streams correspond to $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-values by $S^\flat$ and E^-, and
4. $\lambda\mu_{\text{cons}}$ contexts correspond to $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-terms by $C^\flat$ and $e^\sharp$.

To establish the equational correspondence, we must show that all translations preserve all equalities, and the roundtrip translation of every expression is the same as the original expression up to the respective equational theory.

To begin the correspondence, we consider that the translations preserve equalities. Observe that the $(-)^\flat$, $(-)^\sharp$, and $(-)^-$ translations, as given in Figure 9, are compositional.

Therefore, it suffices to show that the axioms of $\lambda\mu_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ are preserved by each translation. Besides the fact that substitution commutes with translation in either direction, the key property needed is that E^{-} is a valid interpretation of co-values as streams according to $\sharp$-translation. More specifically, one can show by induction on E that for all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-values E:

$$\lambda\mu_{\text{cons}} \vdash E^{\sharp} = [E^{-}]\Box$$

The case for $\mathtt{cdr}(E)$ requires the μ rule and the case for $v \cdot E$ requires the assoc rule. With this fact, the interderivability of the $\lambda\mu_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ axioms follows by routine calculation.

To finish the correspondence, we consider the roundtrip translations. First, note that the roundtrip from $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ to $\lambda\mu_{\text{cons}}$ and back is a provable equality. In particular, observe that the following three properties hold by mutual induction on commands, terms, and co-terms:

1. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ commands c, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (c^{\sharp})^{\flat} = c$.
2. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ terms v, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (v^{\sharp})^{\flat} = v$.
3. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-terms e and $\lambda\mu_{\text{cons}}$ terms v, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (e^{\sharp}[v])^{\flat} = \langle v^{\flat} \| e\rangle$.

The third property is generalized from the usual form of roundtrip translation, and additionally expresses the fact that co-terms lost in a context can always be rediscovered no matter what fills them. From the third property, we get the desired roundtrip equality of co-terms: [1]

$$\text{for all } \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \text{ co-terms } e,\ \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (e^{\sharp})^{\flat} = e \quad .$$

This follows from the translation of contexts in the $\lambda\mu_{\text{cons}}$-calculus into $\tilde{\mu}$-abstractions in the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$-calculus along with the $\eta_{\tilde{\mu}}$ axiom:

$$\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (e^{\sharp})^{\flat} = \tilde{\mu}x.(e^{\sharp}[x])^{\flat} =_{3} \tilde{\mu}x.\langle x \| e\rangle =_{\eta_{\tilde{\mu}}} e \quad .$$

We can also derive a tighter roundtrip equality for establishing the correspondence between co-values and streams:

$$\text{for all } \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \text{ co-values } E,\ \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (E^{-})^{\flat} = E \quad .$$

This follows by $\flat$-translating the equality $\lambda\mu_{\text{cons}} \vdash E^{\sharp}[x] = [E^{-}]x$ (where x is not free in E) which can be composed with the instance of co-term roundtrip equality for co-values: $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash \langle x \| E\rangle =_{3} (E^{\sharp}[x])^{\flat} = ([E^{-}]x)^{\flat} \triangleq \langle x \| (E^{-})^{\flat}\rangle$. Thus, by the $\eta_{\tilde{\mu}}$ axiom, we have $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (E^{-})^{\flat} =_{\eta_{\tilde{\mu}}} \tilde{\mu}x.\langle x \| (E^{-})^{\flat}\rangle = \tilde{\mu}x.\langle x \| E\rangle =_{\eta_{\tilde{\mu}}} E$.

Second, we note that the roundtrip from $\lambda\mu_{\text{cons}}$ to $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ and back is a provable equality. In particular, observe that the following three properties hold by mutual induction on commands, terms, and streams:

1. For all $\lambda\mu_{\text{cons}}$ commands c, $\lambda\mu_{\text{cons}} \vdash (c^{\flat})^{\sharp} = c$.
2. For all $\lambda\mu_{\text{cons}}$ terms v, $\lambda\mu_{\text{cons}} \vdash (v^{\flat})^{\sharp} = v$.
3. For all $\lambda\mu_{\text{cons}}$ streams S, $\lambda\mu_{\text{cons}} \vdash (S^{\flat})^{-} = S$.

[1] Note that we use the syntax, $T \vdash t =_{r} t'$ to indicate that $T \vdash t = t'$ by the rule named r. Similarly, here we use $=_{3}$ to indicate that the equality is true for reason of property 3 above.

The third property relies on the auxiliary injection of co-values into streams to maintain a tighter roundtrip equality which avoids losing the representation of co-terms into contexts caused by the $\sharp$-translation. However, as a corollary of the third property, we also get the fact that $\flat$ and $\sharp$ are inverses for $\lambda\mu_{\text{cons}}$ streams when they are considered as contexts, up to equality:

$$\text{for all } \lambda\mu_{\text{cons}} \text{ streams } S,\ \lambda\mu_{\text{cons}} \vdash (S^{\flat})^{\sharp} = [S]\Box \quad ,$$

which holds by the previously noted fact that $\lambda\mu_{\text{cons}} \vdash E^{\sharp} = [E^{-}]\Box$. Additionally, the first property establishes the fact that $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-terms are in exact correspondence to $\lambda\mu_{\text{cons}}$ contexts by their $\flat$-translation as $\tilde{\mu}$-abstractions. Specifically,

$$\text{for all } \lambda\mu_{\text{cons}} \text{ contexts } C,\ \lambda\mu_{\text{cons}} \vdash (C^{\flat})^{\sharp} = C \quad ,$$

which is provable by the $\lambda\mu_{\text{cons}}$ equational theory of contexts:

$$\lambda\mu_{\text{cons}} \vdash C^{\flat\sharp} \triangleq (\tilde{\mu}x.(C[x])^{\flat})^{\sharp} \triangleq [\delta]((\lambda x.\mu\delta.(C[x])^{\flat\sharp})\ \Box) =_1 [\delta]((\lambda x.\mu\delta.C[x])\ \Box) = C$$

■

4 Equivalent views of functions: pattern matching and projection

The use of `car` and `cdr` in function reduction appear so different from the usual treatment of functions that it might come as a surprise. The rules are justified by the previously established call-by-name continuation-passing style transformation using surjective pairs (Hofmann & Streicher, 2002) as well as a stream model (Nakazawa & Nagai, 2014). However, they can also be understood as projection operations defined in terms of pattern matching (Herbelin, 2005; Munch-Maccagnoni, 2013) according to the macro expansions

$$\mathtt{car}(e) \triangleq \mu\alpha.\langle\mu[(x\cdot\_).\langle x\|\alpha\rangle]\|e\rangle \qquad\qquad \mathtt{cdr}(e) \triangleq \tilde{\mu}x.\langle\mu[(\_\cdot\alpha).\langle x\|\alpha\rangle]\|e\rangle$$

similar to the way that the `fst` and `snd` projections out of a tuple can be defined by pattern matching. These definitions give rise to an equational correspondence between the $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ calculus and the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ equational theory.

The only major complication in establishing the correspondence is that the two languages do not quite share the same notion of co-value. In particular, $\mathtt{cdr}(E)$ is a co-value in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ but its definition in $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ is not a co-value because `cdr` expands into a non-trivial $\tilde{\mu}$-abstraction. However, even though $\mathtt{cdr}(E)$ is not a syntactic co-value in the $\mu\tilde{\mu}^{\rightarrow}_{\eta}$-calculus, it is still a semantic co-value since it behaves like one in the $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ equational theory. The only axiom of the $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ equational theory that mentions co-values is the μ-axiom, which is derivable for the expansion of $\mathtt{cdr}(\alpha)$:

$$\mu\tilde{\mu}^{\rightarrow}_{\eta} \vdash \langle\mu\beta.c\|\mathtt{cdr}(\alpha)\rangle = c[\mathtt{cdr}(\alpha)/\beta] \tag{3}$$

This above equality is the lynchpin that lets us bridge the two different notions of co-value,

$$\langle v \| e \rangle^\circ = \langle v^\circ \| e^\circ \rangle$$

$$\begin{aligned}
\alpha^\circ &\triangleq \alpha & x^\circ &\triangleq x \\
(v \cdot e)^\circ &\triangleq v^\circ \cdot e^\circ & \mu[(x \cdot \alpha).c]^\circ &\triangleq \mu[(x \cdot \alpha).c^\circ] \\
(\tilde{\mu} x.c)^\circ &\triangleq \tilde{\mu} x.(c^\circ) & (\mu \alpha.c)^\circ &\triangleq \mu \alpha.(c^\circ) \\
\mathtt{cdr}(e)^\circ &\triangleq \tilde{\mu} x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha \rangle] \| e^\circ \rangle & \mathtt{car}(e)^\circ &\triangleq \mu \alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha \rangle] \| e^\circ \rangle
\end{aligned}$$

Fig. 10. Translation from $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ to $\mu\tilde{\mu}^{\rightarrow}_{\eta}$

and build an equational correspondence between $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$.

Theorem 2
$\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ are in equational correspondence.

Proof
The translation from $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ into $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ is syntactic inclusion, and the translation from $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ to $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ is the full macro expansion of `car` and `cdr` as given in Figure 10.

The only significant obstacle in showing that macro expansion maps equalities of $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ to equalities of $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ is that the expansion of a $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ co-value, E°, is not always a $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ co-value due to the fact that the expansion of `cdr` is never a co-value. Thus, the μ-axiom of $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ does not map directly to the μ-axiom of $\mu\tilde{\mu}^{\rightarrow}_{\eta}$. However, it turns out that every $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ co-value E still behaves like a co-value in $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ as justified by the extended μ-rule:

$$\mu\tilde{\mu}^{\rightarrow}_{\eta} \vdash \langle \mu \alpha.c \| E^\circ \rangle = c[E^\circ/\alpha]$$

which can be shown by induction on E, using Equation 3 in the `cdr` case. With this derived equality, and the fact that the translation is compositional, it is straightforward to check that the equalities of $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ are interderivable by checking each axiom of each equational theory under translation. More specifically, since many of the axioms are the same, and $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ is a syntactic subset of $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$, it suffices to show that the β- and η-axioms of $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ are derivable in $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$, and that the `car`, cdr, `surj`, `exp`, and ς-family of axioms from $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ are derivable in $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ by their macro expansions.

The last part of the equational correspondence is to show that all roundtrip translations are equalities in their respective theories. For the roundtrip from $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ to $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ and back, this is trivial since $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ is included as a syntactic subset of $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ which is unchanged on the translation back to $\mu\tilde{\mu}^{\rightarrow}_{\eta}$. For the roundtrip from $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ to $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and back, it suffices to observe that the macro expansions of `car` and `cdr` are provable equalities in $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$, as all other syntactic constructs roundtrip to themselves exactly. ■

5 Confluence for extensional call-by-name reduction

Now we return to our original question of confluence as it applies to the $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$ reduction theory. The main obstacle we must face is the fact that the surjectivity rule for call stacks, `surj`, is not left-linear since it requires two projections out of the *same* co-value. This is a

$$\begin{aligned}
c \in \text{Commands} &::= \langle v \| e \rangle \\
v \in \text{Terms} &::= \mu\alpha.c \mid x \mid \mathtt{car}(E) \mid \mu[(x \cdot \alpha).c] \\
E \in \text{Co-Values} &::= \alpha \mid \mathtt{cdr}(E) \mid v \cdot E \\
e \in \text{Co-Terms} &::= \tilde{\mu}x.c \mid E
\end{aligned}$$

Fig. 11. $\mu\tilde{\mu}^{F}_{\text{cons}}$: the focalized sub-syntax of $\mu\tilde{\mu}^{\to}_{\text{cons}}$

problem because we might reduce one of the sub co-values in a `surj`-redex, as in:

$$\begin{tikzcd}
\mathtt{car}(E) \cdot \mathtt{cdr}(E) \arrow[r, "\mathtt{surj}"] \arrow[d] & E \\
\mathtt{car}(E') \cdot \mathtt{cdr}(E) \arrow[r, "\mathtt{surj}", "/" marking] & {}
\end{tikzcd}$$

The two copies of E have gotten out of synch, so inner reductions can destroy surrounding `surj` redexes. As a consequence, many standard rewriting techniques for establishing confluence—such as parallel reduction, full development, and commutation or reordering of reductions—do not directly apply to the $\mu\tilde{\mu}^{\to}_{\text{cons}}$ calculus.

Instead, we look to simplify the calculus, eliminating any extraneous features and keeping only the essential kernel that is necessary for performing computation, until the non-left-linearity of `surj` is no longer problematic. Then, we hoist confluence of the kernel into confluence of the full $\mu\tilde{\mu}^{\to}_{\text{cons}}$-calculus. As it turns out, a minor restriction of the kernel calculus has appeared before as the stack calculus (Carraro *et al.*, 2012), which is already known to be confluent. In repeating the proof of confluence for the extended stack calculus, we take the opportunity to emphasize what we believe to be the single key idea for confluence of both $\Lambda\mu_{\mathtt{cons}}$ and the stack calculus (Nakazawa & Nagai, 2014; Carraro *et al.*, 2012). The confluence of $\mu\tilde{\mu}^{\to}_{\text{cons}}$ via its reduction into the extended stack calculus is new.

In order to relate the original reduction theory of $\mu\tilde{\mu}^{\to}_{\text{cons}}$ with the simpler one of the stack calculus, we will use the directed analog of equational correspondence for reductions known as a Galois connection or an adjunction (Sabry & Wadler, 1997). The conditions of a Galois connection are essentially the same as the four conditions of an equational correspondence, except that we need to be careful about the direction of arrows. More specifically, given a source calculus S and target T, the translations $\flat : S \to T$ and $\sharp : T \to S$ form a *Galois connection* from S to T if and only if the following four conditions hold:

1. ($\flat$) For all terms s_1, s_2 of S, $s_1 \twoheadrightarrow_S s_2$ implies $s_1^\flat \twoheadrightarrow_T s_2^\flat$.
2. ($\sharp$) For all terms t_1, t_2 of T, $t_1 \twoheadrightarrow_T t_2$ implies $t_1^\sharp \twoheadrightarrow_S t_1^\sharp$.
3. ($\flat\sharp$) For all terms s of S, $s \twoheadrightarrow_S (s^\flat)^\sharp$.
4. ($\sharp\flat$) For all terms t of T, $(t^\sharp)^\flat \twoheadrightarrow_T t$.

Additionally, if the fourth condition is strengthened so that any term t of T is syntactically equal to $(t^\sharp)^\flat$, then $\flat$ and $\sharp$ form a *reflection* in S of T. Galois connections are convenient to work with because they compose: given Galois connections from S_1 to S_2 and from S_2 to S_3, we have one from S_1 to S_3. This lets us break a complex translation down into simpler steps and put them back together in the end. More crucially for our purposes, a Galois connection allows us to hoist confluence of the target reduction theory into the source.

Theorem 3
Given a Galois connection from S to T, S is confluent whenever T is. Furthermore, given a reflection in S of T, S is confluent if and only if T is.

Proof
Let $\flat : S \to T$ and $\sharp : T \to S$ be the translations of the Galois connection. Supposing that $s_1 \twoheadleftarrow_S s \twoheadrightarrow_S s_2$, the following diagram commutes by confluence of T:

$$\begin{tikzcd}
 & & & s \arrow[dlll, two heads, "S"'] \arrow[d, "\flat"] \arrow[drrr, two heads, "S"] & & & \\
 & & (\flat) & s^\flat \arrow[dll, two heads, "T"'] \arrow[drr, two heads, "T"] & (\flat) & & \\
s_1 \arrow[r, "\flat"] \arrow[dr, two heads, "S"'] \arrow[dr, phantom, "(\flat\sharp)", near start] & s_1^\flat \arrow[d, "\sharp"] \arrow[drr, two heads, "T"] & & \mathit{confl.} & & s_2^\flat \arrow[d, "\sharp"'] \arrow[dll, two heads, "T"'] & s_2 \arrow[l, "\flat"'] \arrow[dl, two heads, "S"] \arrow[dl, phantom, "(\flat\sharp)"', near start] \\
 & s_1^{\flat\sharp} \arrow[drr, two heads, "S"'] & (\sharp) & t \arrow[d, "\sharp"] & (\sharp) & s_2^{\flat\sharp} \arrow[dll, two heads, "S"] & \\
 & & & t^\sharp & & &
\end{tikzcd}$$

Additionally, a reflection gives us the fact that for any term t of T, $t \equiv (t^\sharp)^\flat$ (where $\equiv$ is syntactic equality), meaning that $t \twoheadrightarrow_T (t^\sharp)^\flat$ by reflexivity. Thus, with a reflection we can swap S with T and $\flat$ with $\sharp$ in the above diagram so that it commutes again by confluence of S. ■

Our proof of confluence for the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$-calculus is broken down into two separate parts:

1. We establish a reflection in the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$-calculus of an extension of the stack calculus (called the Σ^x-calculus in Figure 12). This reflection is formed as a result of normalization in two steps. First, we show that ς-normalization forms a reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of its focalized sub-syntax (called $\mu\tilde{\mu}^{F}_{\text{cons}}$ in Figure 11). Second, we show that $\tilde{\mu}$`exp`-normalization forms a reflection in $\mu\tilde{\mu}^{F}_{\text{cons}}$ of Σ^x. The full reflection comes out from composition of these two steps.
2. We elaborate the confluence proof of the Σ^x-calculus. First, we show that the Σ^x reduction theory is equivalent to one with a restricted `surj` surjectivity rule. The restricted rule only applies to co-values with no other possible reductions, so it avoids the problem where a `surj` redex is destroyed by getting out of sync. Second, we finish the proof by showing that the reduction theory with this restricted `surj` rule is confluent, meaning that the original Σ^x-calculus is also confluent.

To conclude, since $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ contains a reflection of the confluent Σ^x-calculus, from the above theorem $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ must be confluent as well.

5.1 A stack calculus and its reflection

We demonstrate a reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of Σ^x by a sequence of normal forms with respect to two well-behaved subsets of the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ reduction theory. On their own, these sets of reductions are normalizing and confluent, so their normal forms can be computed all at once ahead of time to eliminate particular features of the source $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$-calculus. Furthermore, their normal forms are closed under reduction, so that further reduction does not re-introduce those eliminated features, giving a smaller target calculus. As such, these reductions can be seen as being "administrative" in nature, since they simplify some feature down to more primitive components, resulting in a simpler target calculus.

Reflection by normalization is a rather specific form of a general Galois connection, so we have some extra knowledge about how the source and target calculi are related to one another. In particular, we begin with the reduction theory for the source calculus (S) and divide it into two parts: a set of administrative reductions done upfront during normalization (A), and the remaining non-administrative reductions that are carried over into the target (T). The target calculus then becomes T-reductions over A-normal forms. So long as the A reduction theory is confluent, a reflection in S of T via A-normalization just tells us that full A-normalization commutes across T-reduction. In the following diagrams, a dashed arrow stands for the existence of such a reduction.

Theorem 4
Let S, T, and A be reduction theories such that A is confluent and normalizing and S is equivalent to the (reflexive, transitive) union of T and A. A-normalization forms a reflection in S of T if and only if A-normalization commutes across T reduction, i.e. for all $s_1 \twoheadrightarrow_T s_2$ and their A-normal forms t_1 and t_2, respectively:

$$\begin{tikzcd}
s_1 \arrow[r, two heads, "T"] \arrow[d, two heads, "A"] & s_2 \arrow[d, two heads, "A"] \\
t_1 \arrow[r, two heads, dashed, "T"] & t_2
\end{tikzcd}$$

Proof
First, we note that every term s of S has a unique A-normal form (there is at least one because A is normalizing and at most one because A is confluent) which we denote s^A, so our translation functions are A-normalization ($s^\flat = s^A$) and inclusion of A-normal forms inside the original language S ($t^\sharp = t$). To show the right-to-left implication, given the above commutation, A-normalization and inclusion form a reflection in S of T:

1. ($\flat$): Suppose that $s_1 \twoheadrightarrow_S s_2$. Because S reduction is equivalent to the reflexive, transitive closure over both A and T reductions, we equivalently have that

$$s_1 \twoheadrightarrow_A \twoheadrightarrow_T \twoheadrightarrow_A \twoheadrightarrow_T \ldots \twoheadrightarrow_A \twoheadrightarrow_T s_2$$

We can therefore show that $s_1^A \twoheadrightarrow_T s_2^A$ by induction over the reduction over n in $s_1(\twoheadrightarrow_A \twoheadrightarrow_T)^n s_2$. If $n = 0$ then the result is immediate by reflexivity of T. Otherwise, if $n = 1 + m$ we have $s_1 \twoheadrightarrow_A s_1' \twoheadrightarrow_T s_2'(\twoheadrightarrow_A \twoheadrightarrow_T)^m s_2$, and the result follows from confluence of A, commutation of T over A-normalization, and the inductive hypothesis:

$$\begin{tikzcd}
s_1 \arrow[r, two heads, "A"] \arrow[d, two heads, "A"] \arrow[dr, phantom, "confl."] & s_1' \arrow[r, two heads, "T"] \arrow[d, two heads, "A"] \arrow[dr, phantom, "comm."] & s_2' \arrow[r, two heads, "(\twoheadrightarrow_A \twoheadrightarrow_T)"] \arrow[d, two heads, "A"] \arrow[dr, phantom, "IH"] & s_2 \arrow[d, two heads, "A"] \\
s_1^A \arrow[r, equal] & s_1'^A \arrow[r, two heads, dashed, "T"] & s_2'^A \arrow[r, two heads, dashed, "T"] & s_2^A
\end{tikzcd}$$

2. ($\sharp$): For all A-normal forms t_1 and t_2, $t_1 \twoheadrightarrow_T t_2$ implies $t_1 \twoheadrightarrow_S t_2$ because T reduction is included in S.
3. ($\flat\sharp$): For all s, $s \twoheadrightarrow_S s^A$ because A reduction is included in S.
4. ($\sharp\flat$): For all A-normal forms t, $t^A \equiv t$.

To show the left-to-right implication, given that A-normalization forms a reflection in S of T, the commutation of A-normalization across T reduction is a special case of the first property ($\flat$), since T reduction is included in S. Specifically, if $s_1 \twoheadrightarrow_T s_2$ then since T is included in S, $s_1 \twoheadrightarrow_S s_2$ and so by the property $\flat$, $t_1 \twoheadrightarrow_T t_2$ where t_1 and t_2 are the A-normal forms of s_1 and s_2 respectively. ■

We now build our reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of Σ^x by $\varsigma\tilde{\mu}\texttt{exp}$-normalization in two steps. First, we fully normalize commands, terms and co-terms of the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ by the ς-rules. Second, we normalize ς-normal forms by the $\tilde{\mu}$ and $\texttt{exp}$ rules. We separate these two steps due to the unnecessary complication of performing full $\varsigma\tilde{\mu}\texttt{exp}$-normalization at once: the normal forms are difficult to identify, and even showing that reduction is normalizing is not obvious. The complication is due to the fact that ς-reduction creates *new* $\tilde{\mu}$-abstractions, and thus new $\tilde{\mu}$-redexes. However, when taken separately, ς-normalization produces all the necessary extra $\tilde{\mu}$-abstractions first, so that $\tilde{\mu}$-normalization can easily eliminate them all afterward. And since reflections compose, these two steps can be performed in sequence to build an overall reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of $\varsigma\tilde{\mu}\texttt{exp}$-normal forms.

The ς-normal forms give the focalized sub-syntax of the $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$-calculus, called $\mu\tilde{\mu}^{F}_{\text{cons}}$ in Figure 11, where call stacks are built out of co-values, and stack projections only apply to co-values. In effect, the focalized sub-syntax limits general co-terms, so that a $\tilde{\mu}$-abstraction can only appear at the top of a co-term, and not arbitrarily nested inside call stacks. Also notice that the sub-syntax of $\mu\tilde{\mu}^{F}_{\text{cons}}$ is closed under reduction: once we have fully applied all ς-rules, they never come up again during reduction. Thus, the reduction theory of the $\mu\tilde{\mu}^{F}_{\text{cons}}$-calculus consists of all non-ς rules. This gives us a reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of $\mu\tilde{\mu}^{F}_{\text{cons}}$ by ς-normalization.

Lemma 1
ς-normalization forms a reflection in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ of $\mu\tilde{\mu}^{F}_{\text{cons}}$.

Proof
First, we note that ς-reduction is confluent (it is an orthogonal combinatory reduction system Klop, van Oostrom & van Raamsdonk, 1993)) and normalizing (each application of ς-reduction decreases the number of non-co-values, e, sitting in a call stack, $v \cdot e$, or projection, $\texttt{car}(e)$ or $\texttt{cdr}(e)$). Therefore, by Theorem 4, we only need to show that all other reductions of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$, namely those of $\mu\tilde{\mu}^{F}_{\text{cons}}$, commute over ς-normalization, where we denote the ς-normal form of c as c^{ς} (similarly v^{ς}, e^{ς}, and E^{ς}). In order to establish commutation, we consider commutation of a single $\mu\tilde{\mu}^{F}_{\text{cons}}$ step over ς-normalization:

$$\begin{tikzcd}
c_1 \arrow[r, "\mu\tilde{\mu}^{F}_{\text{cons}}"] \arrow[d, two heads, "\varsigma"] & c_2 \arrow[d, two heads, "\varsigma"] \\
c_1^{\varsigma} \arrow[r, dashed, two heads, "\mu\tilde{\mu}^{F}_{\text{cons}}"] & c_2^{\varsigma}
\end{tikzcd}$$

$$
\begin{aligned}
c \in \text{Commands} &::= \langle v \| E \rangle \\
v \in \text{Terms} &::= \mu\alpha.c \mid x \mid \mathtt{car}(E) \\
E \in \text{Co-Values} &::= \alpha \mid \mathtt{cdr}(E) \mid v \cdot E \\
e \in \text{Co-Terms} &::= \tilde{\mu}x.c \mid E
\end{aligned}
$$

$$
\begin{aligned}
\langle \mu\alpha.c \| E \rangle &\to_{\mu} c[E/\alpha] \\
\mu\alpha.\langle v \| \alpha \rangle &\to_{\eta_\mu} v \\
\tilde{\mu}x.\langle x \| e \rangle &\to_{\eta_{\tilde{\mu}}} e \\
\mathtt{car}(E) \cdot \mathtt{cdr}(E) &\to_{\mathtt{surj}} E \\
\mathtt{car}(v \cdot E) &\to_{\mathtt{car}} v \\
\mathtt{cdr}(v \cdot E) &\to_{\mathtt{cdr}} E
\end{aligned}
$$

Fig. 12. Stack calculus with free variables - Σ^x

from which the full commutation result is obtained by composition over multiple $\mu\tilde{\mu}^F_{\text{cons}}$ steps. The single-step commutation can be shown by mutual induction on commands, terms, and co-terms, considering the possible reductions in each case. Most of these follow directly by the inductive hypothesis, using the additional facts that co-values are closed under reduction and ς-normalization commutes with substitution.

The interesting cases are those in which internal reductions are capable of destroying surrounding redexes. In particular, a `surj` redex can be ruined by putting the co-values out of synch due to asymmetric reduction. Fortunately, because we are forced to fully reduce to the unique ς-normal form, and because co-values are closed under reduction, `surj` reduction commutes:

$$\begin{tikzcd}
\mathtt{car}(E) \cdot \mathtt{cdr}(E) \arrow[r, "\mathtt{surj}"] \arrow[d, two heads, "\varsigma"] & E \arrow[d, two heads, "\varsigma"] \\
\mathtt{car}(E^{\varsigma}) \cdot \mathtt{cdr}(E^{\varsigma}) \arrow[r, dashed, "\mathtt{surj}"] & E^{\varsigma}
\end{tikzcd}$$

The other case in which an internal reduction may destroy an outer redex is in the case of ς rules themselves. This is because a non-co-value co-term may be converted into a co-value by an $\eta_{\tilde{\mu}}$-reduction, which prevents the ς-family of rules from applying and likewise changes the final shape of the ς-normal form. However, substitution by $\tilde{\mu}$ is capable of undoing such an unnecessary ς-reduction, and additional η_μ and $\eta_{\tilde{\mu}}$ reductions clean up the leftover μ- and $\tilde{\mu}$-abstractions:

$$\begin{tikzcd}
v \cdot e \arrow[r, "\eta_{\tilde{\mu}}"] \arrow[d, two heads, "\varsigma"] & v \cdot E \arrow[d, two heads, "\varsigma"] \\
\tilde{\mu}x.\langle \mu\alpha.\langle x \| v^{\varsigma} \cdot \alpha \rangle \| e^{\varsigma} \rangle \arrow[r, dashed, two heads, "\mu\tilde{\mu}^F_{\text{cons}}"] & v^{\varsigma} \cdot E^{\varsigma}
\end{tikzcd}$$

Similar diagrams hold for the other ς rules for $\mathtt{car}(e)$ and $\mathtt{cdr}(e)$ when e is a non-co-value that reduces to a co-value. ■

The second step of our reflection is to normalize the focalized $\mu\tilde{\mu}^F_{\text{cons}}$ sub-syntax by $\tilde{\mu}$`exp`-reduction. These normal forms are exactly the stack calculus (Carraro *et al.*, 2012) extended with free variables, Σ^x, shown in Figure 12. From the typed perspective, as in

$$\frac{(x:A)\in\Gamma}{\Gamma\vdash x:A\mid\Delta}[id_R] \qquad \frac{(\alpha:A)\in\Delta}{\Gamma\mid\alpha:A\vdash\Delta}[id_L]$$

$$\frac{c:(\Gamma\vdash\alpha:A,\Delta)}{\Gamma\vdash\mu\alpha.c:A\vdash\Delta}[Act_R] \qquad \frac{c:(\Gamma,x:A\vdash\Delta)}{\Gamma\mid\tilde{\mu}x.c:A\vdash\Delta}[Act_L]$$

$$\frac{\Gamma\vdash v:A\mid\Delta \qquad \Gamma\mid E:A\vdash\Delta}{\langle v\|E\rangle:(\Gamma\vdash\Delta)}[cut]$$

$$\frac{\Gamma\vdash v:A\mid\Delta \qquad \Gamma\mid E:B\vdash\Delta}{\Gamma\mid v\cdot E:A\to B\vdash\Delta}[\to Intro]$$

$$\frac{\Gamma\mid E:A\to B\vdash\Delta}{\Gamma\vdash \mathtt{car}(E):A\mid\Delta}[\to Elim_1] \qquad \frac{\Gamma\mid E:A\to B\vdash\Delta}{\Gamma\mid \mathtt{cdr}(E):B\vdash\Delta}[\to Elim_2]$$

Fig. 13. Simple Type Assignment for Σ^x

Figure 13, has only elimination rules and left introduction rules. Together with natural deduction (only right rules) and the pure sequent calculus we started with (only introduction rules on both sides) it thus represents one of a multitude of choices for a logical inference system. But, as (Carraro *et al.*, 2012) noticed, this choice is interesting computationally. In effect, $\tilde{\mu}$exp-normalization eliminates all variable binders within commands, terms, and co-values, replacing them either by substitution or by a projection out of a co-variable. Therefore, commands, terms, and co-values do not contain *any* variable binders. The one technical detail is the presence of general co-terms remaining in the syntax. This is necessary to form a reflection in $\mu\tilde{\mu}^F_{\text{cons}}$ because of the fact that a given co-term $\tilde{\mu}x.c$ will still reduce to another $\tilde{\mu}$-abstraction $\tilde{\mu}x.c'$. However, the only $\tilde{\mu}$-abstraction in the resulting co-term will be the one at the top; c' contains no other $\tilde{\mu}$-abstractions. Thus, besides the possibility of one $\tilde{\mu}$-abstraction at the very top of a co-term, the Σ^x-calculus has no variable binders. And if general co-terms are not of interest, this detail may be elided. Furthermore, like the focalized $\mu\tilde{\mu}^F_{\text{cons}}$ sub-syntax, the syntax of the Σ^x-calculus is also closed under reduction, so the reduction theory of the Σ^x-calculus only consists of the `car`, `cdr`, `surj`, μ, η_μ, and $\eta_{\tilde{\mu}}$ (at the top of a co-term only) rules. This gives us a reflection in $\mu\tilde{\mu}^F_{\text{cons}}$ of Σ^x by $\tilde{\mu}$exp-normalization.

Lemma 2
$\tilde{\mu}$exp-normalization forms a reflection in $\mu\tilde{\mu}^F_{\text{cons}}$ of Σ^x.

Proof
First, we note that $\tilde{\mu}$exp-reduction is confluent since it is an orthogonal combinatory reduction system. Additionally, it is normalizing since, even though $\tilde{\mu}$ reduction can duplicate the number of $\tilde{\mu}$exp-redexes, no *new* redexes are created. Therefore, by Theorem 4, we only need to show that all other reductions of $\mu\tilde{\mu}^F_{\text{cons}}$, namely those of Σ^x, commute over $\tilde{\mu}$exp-normalization. This follows analogously to the proof of Lemma 1, where again the non-left-linear nature of `surj` does not interfere with commutation because we are forced to perform full $\tilde{\mu}$exp-normalization. ■

Theorem 5
$\varsigma\tilde{\mu}$exp-normalization forms a reflection in $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ of Σ^x.

Proof
By composition of the reflections in $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ of $\mu\tilde{\mu}_{\text{cons}}^{F}$ (Lemma 1) and in $\mu\tilde{\mu}_{\text{cons}}^{F}$ of Σ^x (Lemma 2). ■

5.2 Confluence of a stack calculus

Now we focus on establishing confluence of the Σ^x-calculus. The Σ^x-calculus is simpler than the full $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$-calculus, doing away with several constructs and reductions. However, the Σ^x reduction theory still has the complication of the `surj` rule, whose non-left linearity defeats many approaches of establishing confluence.

Surprisingly, we can completely side-step the non-left-linearity problem of surjective call stacks by restricting the `surj` rule. Instead of matching general co-values, we will only bring together a certain form of stuck co-values consisting of a chain of `cdr` projections out of a co-variable (where $\mathtt{cdr}^n(\alpha)$ means n applications of `cdr` to α):

$$\mathtt{car}(\mathtt{cdr}^n(\alpha)) \cdot \mathtt{cdr}(\mathtt{cdr}^n(\alpha)) \rightarrow_{\mathtt{surj}'} \mathtt{cdr}^n(\alpha)$$

The restricted $\mathtt{surj}'$ rule is clearly a particular instance of the more general `surj` rule, and replacing `surj` with $\mathtt{surj}'$ gives us the simplified Σ'^x reduction theory. However, $\mathtt{surj}'$ identifies an application of surjectivity that cannot get out of synch, since $\mathtt{cdr}^n(\alpha)$ is a normal form that cannot reduce further. Therefore, the *only* possible reduct of $\mathtt{car}(\mathtt{cdr}^n(\alpha)) \cdot \mathtt{cdr}(\mathtt{cdr}^n(\alpha))$ is $\mathtt{cdr}^n(\alpha)$, and so $\mathtt{surj}'$ redexes cannot be destroyed by other reductions. This side-steps the problematic aspect of `surj` that we began with. Unfortunately, $\mathtt{surj}'$ brings up a different problem: $\mathtt{surj}'$ redexes are not closed under substitution, so *surrounding* μ reductions naïvely destroy inner $\mathtt{surj}'$ reduction. Miraculously though, the other reduction rules pick up the slack when $\mathtt{surj}'$ fails, and so Σ^x and Σ'^x end up being equivalent reduction theories.

Lemma 3
The Σ^x and Σ'^x reduction theories are equivalent: $c \twoheadrightarrow_{\Sigma^x} c'$ if and only if $c \twoheadrightarrow_{\Sigma'^x} c'$, and likewise for (co-)terms. Furthermore, the `carcdrsurj` and $\mathtt{carcdrsurj}'$ reduction theories are equivalent.

Proof
The only difference between the two reduction theories is the rule for surjectivity of call stacks, `surj` in Σ^x versus the restricted $\mathtt{surj}'$ in Σ'^x. The $\mathtt{surj}'$ is clearly a particular family of instances of the `surj` rule, $\mathtt{car}(E) \cdot \mathtt{cdr}(E) \rightarrow E$, where E must have the form $\mathtt{cdr}^n(\alpha)$. So Σ'^x is simulated by Σ^x step by step.

To go the other way, we only need to show that the unrestricted `surj` rule is simulated by $\mathtt{carcdrsurj}'$ reduction. To demonstrate the simulation, we consider what happens when we substitute an arbitrary co-value E in for the co-variable of a $\mathtt{surj}'$ redex. In particular, we demonstrate the following reduction holds:

$$\mathtt{car}(\mathtt{cdr}^n(E)) \cdot \mathtt{cdr}(\mathtt{cdr}^n(E)) \twoheadrightarrow_{\mathtt{carcdrsurj}'} \mathtt{cdr}^n(E)$$

Crucially, this reduction holds because E can only be some form of call stack and nothing else, so that the general extensionality reduction is simulated by the `car` and `cdr` computational rules for call stacks. This can be seen by induction on n and considering cases on the possible co-values for E, where we take any `cdr` projections at the top of E to be included in the chain of projections mentioned by the rule:

- If $E = \alpha$, then we can directly apply the $\mathtt{surj}'$ rule:
$$\mathtt{car}(\mathtt{cdr}^n(\alpha)) \cdot \mathtt{cdr}(\mathtt{cdr}^n(\alpha)) \to_{\mathtt{surj}'} \mathtt{cdr}^n(\alpha)$$
- If $E = v \cdot E'$ and $n = 0$ then the simulation follows by `car` and `cdr` reductions:
$$\begin{aligned}\mathtt{car}(\mathtt{cdr}^0(v\cdot E')) \cdot \mathtt{cdr}(\mathtt{cdr}^0(v\cdot E')) &= \mathtt{car}(v\cdot E') \cdot \mathtt{cdr}(v\cdot E') \\ &\to_{\mathtt{car}} v \cdot \mathtt{cdr}(v\cdot E') \to_{\mathtt{cdr}} v \cdot E' = \mathtt{cdr}^0(v\cdot E')\end{aligned}$$
- If $E = v \cdot E'$ and $n = n' + 1$ then the simulation follows by two `cdr` reductions and the inductive hypothesis:
$$\begin{aligned}\mathtt{car}(\mathtt{cdr}^{n+1}(v\cdot E')) \cdot \mathtt{cdr}(\mathtt{cdr}^{n+1}(v\cdot E')) &\to_{\mathtt{cdr}}\to_{\mathtt{cdr}} \mathtt{car}(\mathtt{cdr}^n(E')) \cdot \mathtt{cdr}(\mathtt{cdr}^n(E')) \\ &\twoheadrightarrow_{IH} \mathtt{cdr}^n(E')\end{aligned}$$
- Otherwise, we cannot have $E = \mathtt{cdr}(E')$, because this `cdr` surrounding E' is joined with the chain of `cdr` projections in the rule:
$$\mathtt{car}(\mathtt{cdr}^n(\mathtt{cdr}(E'))) \cdot \mathtt{cdr}(\mathtt{cdr}^n(\mathtt{cdr}(E'))) = \mathtt{car}(\mathtt{cdr}^{n+1}(E')) \cdot \mathtt{cdr}(\mathtt{cdr}^{n+1}(E'))$$

The `surj` rule is the instance of the above reduction where $n = 0$. Thus `carcdrsurj` is simulated by $\mathtt{carcdrsurj}'$, so the two are equivalent. Furthermore, Σ^x is simulated by Σ'^x, so the two are equivalent reduction theories. ■

Now that we have isolated and defeated the non-left-linearity problem of surjective call stacks within the Σ'^x reduction theory, we can show that it is confluent. From this point, the proof of confluence for the simplified Σ'^x theory is entirely routine, where the only technical detail that needs to be addressed is the fact that reduction is closed under substitution, which we already saw in Lemma 3 when equating the Σ^x and Σ'^x reduction theories.

Lemma 4
The Σ'^x reduction theory is confluent.

Proof
We demonstrate confluence by a divide and conquer method, separating the reductions dealing with call stacks (`car`, `cdr`, and $\mathtt{surj}'$) from the reductions dealing with variable binding (μ, η_μ, and $\eta_{\tilde{\mu}}$). Observe that the $\mathtt{carcdrsurj}'$ reduction theory is confluent since it is subcommutative [2]. Additionally, the $\mu\eta_\mu\eta_{\tilde{\mu}}$ reduction theory is confluent since it is an orthogonal combinatory reduction system. We also observe that these two sub theories

[2] By which we mean that critical pairs come together in zero or one step: whenever $t_1 \leftarrow t \to t_2$ there exists some t' such that $t_1 \to^= t' \leftarrow^= t_2$

commute:

$$\begin{tikzcd}
c_1 \arrow[r, two heads, "\mathtt{carcdrsurj}'"] \arrow[d, two heads, "\mu\eta_\mu\eta_{\tilde\mu}"'] & c_1' \arrow[d, dashed, two heads, "\mu\eta_\mu\eta_{\tilde\mu}"] \\
c_2 \arrow[r, dashed, two heads, "\mathtt{carcdrsurj}'"] & c_2'
\end{tikzcd}$$

and similarly for terms and co-terms, which follows by induction from the simpler one-step diagram (where $\rightarrow^{=}$ denotes zero or one steps):

$$\begin{tikzcd}
c_1 \arrow[r, "\mathtt{carcdrsurj}'"] \arrow[d, "\mu\eta_\mu\eta_{\tilde\mu}"'] & c_1' \arrow[d, dashed, "\mu\eta_\mu\eta_{\tilde\mu}", "="'] \\
c_2 \arrow[r, dashed, two heads, "\mathtt{carcdrsurj}'"] & c_2'
\end{tikzcd}$$

Confluence of the whole Σ'^x reduction theory follows from the Hindley-Rosen lemma.[3] The only challenge is to show that the $\mathtt{carcdrsurj}'$ reduction theory is closed under substitution: that $c \twoheadrightarrow_{\mathtt{carcdrsurj}'} c'$ implies $c[E/\alpha] \twoheadrightarrow_{\mathtt{carcdrsurj}'} c'[E/\alpha]$, and so on. This is not a trivial property because of the restricted form of the $\mathtt{surj}'$ rule, whose redexes are destroyed by substitution of co-values for their free co-variable: for example $\mathtt{car}(\alpha) \cdot \mathtt{cdr}(\alpha) \rightarrow_{\mathtt{surj}'} \alpha$ but $(\mathtt{car}(\alpha) \cdot \mathtt{cdr}(\alpha))[x \cdot \beta/\alpha] = \mathtt{car}(x \cdot \beta) \cdot \mathtt{cdr}(x \cdot \beta) \not\rightarrow_{\mathtt{surj}'}$. However, the $\mathtt{carcdrsurj}'$ reduction theory is equivalent to the $\mathtt{carcdrsurj}$ reduction theory (Lemma 3) which *is* closed under substitution, so $\mathtt{carcdrsurj}'$ is closed under substitution as well. As a result, we find that $\mathtt{carcdrsurj}'$ and $\mu\eta_\mu\eta_{\tilde\mu}$ reductions commute as shown above, and so Σ'^x is confluent. ■

Theorem 6
The Σ^x and $\mu\tilde\mu^{\rightarrow}_{\text{cons}}$ reduction theories are confluent.

Proof
Because Σ'^x is confluent (Lemma 4) and Σ'^x and Σ^x are equivalent reduction theories (Lemma 3), then Σ^x is also confluent. Furthermore, because there is a Galois connection from $\mu\tilde\mu^{\rightarrow}_{\text{cons}}$ to Σ^x (Theorem 5) then $\mu\tilde\mu^{\rightarrow}_{\text{cons}}$ is also confluent (Theorem 3). ■

6 Extensionality with other structures and strategies

Extending the treatment of functions discussed here to other structures in programming languages also presents challenges. Consider the negative form of product (&). Since negative products share the same polarity as functions, they also share the same construction/deconstruction bias: co-terms of products are constructed whereas terms pattern match on their context. This gives us two new co-terms of the form $\pi_1(e)$ and $\pi_2(e)$, which can be interpreted as building a context that requests either the first or second component of a product and sends the reply to the context e. On the other side, we have the term $\mu[\pi_1(\alpha_1).c_1|\pi_2(\alpha_2).c_2]$ which expresses the fact that the product is an object waiting for a

[3] If $\twoheadrightarrow_A$ and $\twoheadrightarrow_B$ are two confluent rewriting systems that commute then $\twoheadrightarrow_{A\cup B}$ is confluent.

request. If the first component is requested then c_1 is executed with α_1 bound to the context inside the request message, as described by the following reduction rule:

$$\langle \mu[\pi_1(\alpha_1).c_1 | \pi_2(\alpha_2).c_2] \| \pi_1(e) \rangle \to_{\beta^{\&}} \langle \mu\alpha_1.c_1 \| e \rangle$$

We have a similar rule to handle the case if the second component is requested. The η rule captures the fact that request-forwarding terms can be simplified:

$$\mu[\pi_1(\alpha_1).\langle v \| \pi_1(\alpha_1)\rangle | \pi_2(\alpha_2).\langle v \| \pi_2(\alpha_2)\rangle] \to_{\eta^{\&}} v$$

Predictably, we see the same conflict between extensionality and control that we had with functions, as expressed by the critical pair:

$$\begin{aligned} v_0 = {} & \mu[\pi_1(\alpha).\langle \mu\beta.c \| \pi_1(\alpha)\rangle \\ & \quad |\pi_2(\alpha).\langle \mu\beta.c \| \pi_2(\alpha)\rangle] \\ \mu\beta.c \leftarrow_{\eta^{\&}} v_0 & \twoheadrightarrow_{\mu} \mu[\pi_1(\alpha).c[\pi_1(\alpha)/\beta] | \pi_2(\alpha).c[\pi_2(\alpha)/\beta]] \end{aligned}$$

However, it is far from obvious how to adapt the solution used for functions to work for products as well. The `exp` rule converts a function—a value that decomposes a call stack—into a μ-abstraction. Unfortunately, a product contains *two* branches instead of one, and it is not clear how to merge two arbitrary branches into a single μ-abstraction. We might be inclined to add the following reduction

$$\mu[\pi_1(\alpha).c[\pi_1(\alpha)/\beta] | \pi_2(\alpha).c[\pi_2(\alpha)/\beta]] \to \mu\beta.c$$

for α not occurring in c. However, this rule is suspicious since the pattern $\pi_i(\alpha)$ can easily be destroyed. In fact, a similar rule for functions (which corresponds to the backward ν-rule):

$$\mu[(x \cdot \alpha).c[x \cdot \alpha/\beta] \to \mu\beta.c$$

gives rise to a simple counterexample:

$$\begin{aligned} v_0 = {} & \mu[x\cdot\alpha].\langle \mu[\_\cdot\_].\langle \mu[\_\cdot\_].\langle z \| \delta \rangle \\ & \qquad\qquad\qquad\qquad \| x \cdot \alpha \rangle \\ & \qquad\qquad \| \delta \rangle \\ \mu\beta.\langle \mu[\_\cdot\_].\langle \mu[\_\cdot\_].\langle z \| \delta \rangle \quad & \leftarrow v_0 \twoheadrightarrow_{\mu} [x\cdot\alpha].\langle \mu[\_\cdot\_].\langle z \| \delta \rangle \| \delta \rangle \\ \| \beta \rangle \qquad & \\ \| \delta \rangle \qquad\qquad\qquad & \end{aligned}$$

Adding a positive notion of product, the tensor $\otimes$, is also problematic. On the term side, a positive product is constructed by putting together two terms in the pair (v, v') and deconstructed via the co-term $\tilde{\mu}[(x,y).c]$ which pattern matches on its input term. The β and η rules are :

$$\langle (v,v') \| \tilde{\mu}[(x,y).c] \rangle \to_{\beta^{\otimes}} \langle v \| \tilde{\mu}x.\langle v' \| \tilde{\mu}y.c \rangle\rangle \qquad \tilde{\mu}[(x,y).\langle (x,y) \| E \rangle] \to_{\eta^{\otimes}} E$$

(Since $\tilde{\mu}[(x,y).c]$ is a value in call-by-name, the restriction on the η rule guarantees that a value is not turned into a non-value, similar to the restriction on η for call-by-value functions.) Unfortunately, our proof of confluence relies on co-values always being reducible to simple, normalized structures, containing no reducible sub-commands. However, decomposition of tuples, $\tilde{\mu}[(x,y).c]$, is a co-value which contains arbitrary sub-commands and therefore our proof of confluence does not apply. We conjecture that confluence is lost because in the `surj` redex

$$\mathtt{car}(\tilde{\mu}[(x,y).c]) \cdot \mathtt{cdr}(\tilde{\mu}[(x,y).c])$$

the two occurrences of command c can get out of synch, destroying the `surj` redex, similar to what is happening in Klop's counterexample of confluence for λ-calculus extended with surjective pairing (Klop & de Vrijer, 1989).

Since call-by-value is dual to call-by-name, similar problems and solutions arise in call-by-value calculi along with similar limitations. Consider the tensor product, which has a stronger η-rule (since any co-term is a co-value in call-by-value):

$$\tilde{\mu}[(x,y).\langle (x,y) \| e \rangle] \to_{\eta^{\otimes}} e$$

whereas the β-rule remains the same. As pairs have dual properties to functions, the dual of the counterexample shown in Equation (2) unsurprisingly arises in call-by-value:

$$\tilde{\mu}z.\langle y \| \beta \rangle \leftarrow_{\eta^{\otimes}} \tilde{\mu}[(x,y).\langle (x,y) \| \tilde{\mu}z.\langle y \| \beta \rangle \rangle] \to_{\tilde{\mu}} \tilde{\mu}[(x,y).\langle y \| \beta \rangle]$$

It might help to consider this example in a more natural style from functional programming, where we have the following β- and η-rules for decomposing pairs:[4]

$$\begin{aligned} \mathtt{case}\ (v_1,v_2)\ \mathtt{of}\ (x_1,x_2) \Rightarrow v &\to_{\beta^{\otimes}} \mathtt{let}\ x_1 = v_1\ \mathtt{in}\ \mathtt{let}\ x_2 = v_2\ \mathtt{in}\ v \\ \mathtt{case}\ v\ \mathtt{of}\ (x,y) \Rightarrow E[(x,y)] &\to_{\eta^{\otimes}} E[v] \end{aligned}$$

In this notation, the above critical pair appears as:

$$\mathtt{let}\ z = v\ \mathtt{in}\ w \leftarrow_{\eta^{\otimes}} \mathtt{case}\ v\ \mathtt{of}\ (x,y) \Rightarrow \mathtt{let}\ z = (x,y)\ \mathtt{in}\ w \to_{\mathtt{let}} \mathtt{case}\ v\ \mathtt{of}\ (x,y) \Rightarrow w$$

We can adopt the same solution for call-by-value pairs as we did for call-by-name functions, by converting patterns to projections and adding a surjectivity reduction:

$$\tilde{\mu}[(x,y).c] \to \tilde{\mu}z.c[\mathtt{fst}(z)/x, \mathtt{snd}(z)/y]$$
$$\mathtt{fst}(V_1,V_2) \to V_1 \qquad \mathtt{snd}(V_1,V_2) \to V_2 \qquad (\mathtt{fst}(V),\mathtt{snd}(V)) \to V$$

However, this solution does not scale in similar ways. It is not obvious how to apply this solution to a disjunctive type. Consider the additive sum type $\oplus$ which comes with two new ways of forming terms, $\iota_1(v)$ and $\iota_2(v)$, and a pattern matching co-term $\tilde{\mu}[\iota_1(x).c_1|\iota_2(y).c_2]$ with the reduction rules

$$\langle \iota_i(v) \| \tilde{\mu}[\iota_1(x_1).c_1 \mid \iota_2(x_2).c_2] \rangle \to_{\beta^{\oplus}} c_i[v/x_i]$$
$$\tilde{\mu}[\iota_1(x).\langle \iota_1(x) \| e \rangle \mid \iota_2(x).\langle \iota_2(x) \| e \rangle] \to_{\eta^{\oplus}} e$$

We witness the same counterexample of confluence we had for & in call-by-name. Second, adding functions (which are a co-data type) to the calculus breaks confluence of the surjectivity rule for pairs—a previously known and well-studied problem (Klop & de Vrijer,

[4] Note that the stronger reduction $\mathtt{case}\ v\ \mathtt{of}\ (x,y) \Rightarrow v'[(x,y)/z] \to v'[v/z]$ is not valid in an untyped call-by-value calculus including non-termination. For example, $\mathtt{case}\ \Omega\ \mathtt{of}(x,y) \Rightarrow \lambda d.(x,y)$ loops forever, whereas the reduct $\lambda d.\Omega$ does not under call-by-value evaluation. Thus, we restrict the η-rule for pairs to only apply when the decomposed pair appears in the eye of an evaluation context.

1989)—because values can contain arbitrary reducible sub-terms that can get out of synch:

$$(\mathtt{fst}(\lambda x.v), \mathtt{snd}(\lambda x.v))$$

7 Conclusion

We have seen how the interpretation of functional objects through projections out of their call stack resolves the problems with confluence in a lazy λ-calculus with both control and extensionality. Further, we have shown how that interpretation arises naturally from the standpoint that functions are a co-data type, so the observations of functions deserve just as much attention as the functions themselves. Indeed, as our equational correspondence result makes clear, defining functions by projection adds nothing that wasn't already there in the theory from the beginning. The only trick is noticing that λ-abstractions are not the only way to describe functions, and that η-contraction and -expansion are not the only operational interpretations of the η-law.

The projection-based interpretation of functions can be traced back to the call-by-name continuation-passing style transformation of the λ-calculus that validates η (Hofmann & Streicher, 2002). In continuation-passing style, programs are inverted so that function types are explained in terms of a different type of surjective products. Here, we use the sequent calculus as a vehicle for studying the surjective nature of functions in a more direct style, enabled by the equal consideration given to both producers and consumers. Indeed, the sequent calculus explanation of surjective call stacks does not need to introduce other types to explain functions. As presented here, functions are defined independently in their own right without referencing products or negation, following an orthogonal approach to studying logical connectives (Pfenning, 2002). Furthermore, even though $\lambda\mu_{\text{cons}}$ (Nakazawa & Nagai, 2014) is based on the λ-calculus, its streams are logically interpreted as left rules that come from sequent calculus instead of natural deduction. Therefore, we find that in the same way a symmetric treatment for assuming and concluding facts is important in logic, a symmetric treatment for producing and consuming information is important in programming languages as well.

The effectiveness of the projection-based approach for solving the problems with lazy functions makes it enticing to try to extend it to other systems. However, we see that this technique does not extend easily. An interesting question for future work is to see if there are *other* computational interpretations of extensional axioms which can be used to address the challenge of extensional rewriting for other systems, such as the call-by-value lambda calculus or languages with additional types besides functions.

Extensional reasoning principles like the η-law are fundamentally important for equational reasoning about programs. It is a significant loss if we must give up these principles for reasoning about programs because of the implementation details of program execution. We hope the present work provides a good framework for building up more featureful systems while ensuring that we are not forced to make such a harsh tradeoff.

References

Ariola, Zena M., & Blom, Stefan. (2002). Skew confluence and the lambda calculus with letrec. *Annals of pure and applied logic*, **117**(3), 95 – 168.

Barendregt, H. P. (1984). *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, Amsterdam.

Carraro, Alberto, Ehrhard, Thomas, & Salibra, Antonino. (2012). The stack calculus. *Pages 93–108 of: Workshop on logical and semantic frameworks, with applications*.

Curien, Pierre-Louis, & Herbelin, Hugo. (2000). The duality of computation. *Pages 233–243 of: Proceedings of the international conference on functional programming*.

Curien, Pierre-Louis, & Munch-Maccagnoni, Guillaume. (2010). The duality of computation under focus. *IFIP international conference on theoretical computer science*.

Danvy, Olivier, & Nielsen, Lasse R. (2004). *Refocusing in reduction semantics*. Tech. rept. BRICS.

David, Rene, & Py, Walter. (2001). $\lambda\mu$-calculus and Bohm's theorem. *Journal of symbolic logic*, **66**(1), 407–413.

Downen, Paul, & Ariola, Zena M. (2014). The duality of construction. *Pages 249–269 of: Programming languages and systems - 23rd European symposium on programming, ESOP 2014*.

Fujita, Ken-etsu. (2003). A sound and complete CPS-translation for $\lambda\mu$-calculus. *Pages 120–134 of: Typed lambda calculi and applications*. LNCS, vol. 2701. Springer Berlin Heidelberg.

Herbelin, Hugo. (2005). *C'est maintenant qu'on calcule : Au cœur de la dualité*. Habilitation à diriger les reserches, Université Paris 11.

Herbelin, Hugo, & Ghilezan, Silvia. (2008). An approach to call-by-name delimited continuations. *Pages 383–394 of: Proceedings of the symposium on principles of programming languages*. ACM.

Hofmann, Martin, & Streicher, Thomas. (2002). Completeness of continuation models for $\lambda\mu$-calculus. *Information and computation*, **179**(2), 332 – 355.

Klop, J. W., & de Vrijer, R. C. (1989). Unique normal forms for lambda calculus with surjective pairing. *Information and computation*, **80**(2), 97 – 113.

Krivine, Jean-Louis. (2007). A call-by-name lambda-calculus machine. *Higher-order and symbolic computation*, **20**(3), 199–207.

Munch-Maccagnoni, Guillaume. (2013). *Syntax and Models of a non-Associative Composition of Programs and Proofs*. Ph.D. thesis, Université Paris Diderot.

Munch-Maccagnoni, Guillaume, & Scherer, Gabriel. (2015). Polarised intermediate representation of lambda calculus with sums. *Pages 127–140 of: Logic in computer science (LICS)*.

Nakazawa, Koji, & Nagai, Tomoharu. (2014). Reduction system for extensional lambda-mu calculus. *Pages 349–363 of: Rewriting and typed lambda calculi*. LNCS.

Parigot, Michel. (1992). Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. *Pages 190–201 of: Proceedings of the international conference on logic programming and automated reasoning*. ACM.

Pfenning, Frank. (2002). Logical frameworks—a brief introduction. *Pages 137–166 of: Proof and system-reliability*. NATO Science Series II, vol. 62. Kluwer Academic Publishers.

Pitts, Andrew M. (2000). Parametric polymorphism and operational equivalence. *Mathematical structures in computer science*, **10**(3), 321–359.

Plotkin, G. D. (1975). Call-by-name, call-by-value and the λ-calculus. *Theoretical computer science*, **1**(2), 125 – 159.

Sabry, Amr, & Felleisen, Matthias. (1993). Reasoning about programs in continuation-passing style. *Lisp and symbolic computation*, **6**(3-4), 289–360.

Sabry, Amr, & Wadler, Philip. (1997). A reflection on call-by-value. *ACM transactions on programming languages and systems*, **19**(6), 916–941.

Saurin, Alexis. (2010). Typing streams in the $\Lambda\mu$-calculus. *ACM transactions on computational logic*, **11**(4), 28:1–28:34.

Zeilberger, Noam. (2009). *The logical basis of evaluation order and pattern-matching*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

A Correspondence between natural deduction and sequent calculus

Here, we give the details of Theorem 1, proving auxiliary lemmas as needed.

The equational correspondence between $\lambda\mu_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ involves the syntactic category of $\lambda\mu_{\text{cons}}$ *contexts* (that is, commands with a term-shaped hole), which means that we need to know when contexts are equated. We say that any two such contexts are equal if and only if they are equal commands for every possible filling. More formally, given any two contexts C and C', $\lambda\mu_{\text{cons}} \vdash C = C'$ if and only if for all $\lambda\mu_{\text{cons}}$ terms v, $\lambda\mu_{\text{cons}} \vdash C[v] = C'[v]$. As a lighter notation for equating two contexts, we will denote the universally quantified term of the equality by $\square$, which is just another metavariable for terms when used for this particular purpose, and write C as shorthand for $C[\square]$.

First, we turn to the direct translation for co-values, showing it to be compatible with the $\sharp$ translation given for general co-terms.

Lemma 5
For all co-values E, $\lambda\mu_{\text{cons}} \vdash E^{\sharp} = [E^{-}]\square$.

Proof
By induction on E.

- $$\begin{aligned} \alpha^{\sharp} &\triangleq [\alpha]\square \\ &\triangleq [\alpha^{-}]\square \end{aligned}$$

- $$\begin{aligned} (v\cdot E)^{\sharp} &\triangleq E^{\sharp}[\square\ v^{\sharp}] \\ &= [E^{-}](\square\ v^{\sharp}) && \text{Inductive Hypothesis} \\ &= [v^{\sharp} :: E^{-}]\square && \text{assoc} \\ &\triangleq [(v\cdot E)^{-}]\square \end{aligned}$$

- $$\begin{aligned} \mathrm{cdr}(E)^{\sharp} &= E^{\sharp}[\mu\alpha.[\mathrm{cdr}\ \alpha]\square] \\ &= [E^{-}](\mu\alpha.[\mathrm{cdr}\ \alpha]\square) && \text{Inductive Hypothesis} \\ &= [\mathrm{cdr}\ E^{-}]\square && \mu \\ &\triangleq [\mathrm{cdr}(E)^{-}]\square \end{aligned}$$

■

The next property we show is that the translations are compatible with substitution.

Lemma 6 (*Substitution property for* $\flat$)
For any command, term, or co-term t of $\lambda\mu_{\text{cons}}$, we have that $(t[v/x])^{\flat} \triangleq t^{\flat}[v^{\flat}/x]$ and $(t[S/\alpha])^{\flat} \triangleq t^{\flat}[S^{\flat}/\alpha]$.

Proof
$(-)^{\flat}$ is compositional and so these follow by induction on t. ■

Lemma 7 (*Substitution property for* $\sharp$)
For t being any command, term, or stack of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ then

1. $(t[v/x])^\sharp \triangleq t^\sharp[v^\sharp/x]$ and
2. $\lambda\mu_{\text{cons}} \vdash (t[E/\alpha])^\sharp = t^\sharp[E^-/\alpha]$

Proof
The first point follows because $^\sharp$ is compositional. The second point follows by induction on t, with the non-immediate case being the base case of the variable α where

$$\begin{aligned}(\alpha[E/\alpha])^\sharp &\triangleq E^\sharp \\ &= [E^-]\Box \qquad \text{Lemma 5} \\ &\triangleq ([\alpha]\Box)[E^-/\alpha] \\ &\triangleq \alpha^\sharp[E^-/\alpha]\end{aligned}$$

■

Lemma 8 (β *derivable for* $\lambda\mu_{cons}$)
The following holds for all terms v_1 and v_2

$$\lambda\mu_{\text{cons}} \vdash (\lambda x.v_1)\ v_2 = v_1[v_2/x]$$

Proof
By calculation.

$$\begin{aligned}(\lambda x.v_1)\ v_2 &=_{\eta_\mu} \mu\alpha.[\alpha]((\lambda x.v_1)\ v_2) \\ &=_{\text{assoc}} \mu\alpha.[v_2 :: \alpha](\lambda x.v_1) \\ &=_{\text{exp}} \mu\alpha.[v_2 :: \alpha](\mu\beta.[\texttt{cdr}\ \beta]v_1[(\texttt{car}\ \beta)/x] \\ &=_{\mu} \mu\alpha.[\texttt{cdr}\ (v_1 :: \alpha)]v_1[(\texttt{car}\ (v_2 :: \alpha)/x] \\ &=_{\texttt{cdr}} \mu\alpha.[\alpha]v_1[(\texttt{car}\ (v_2 :: \alpha)/x] \\ &=_{\texttt{car}} \mu\alpha.[\alpha]v_1[v_2/x] \\ &=_{\eta_\mu} v_1[v_2/x]\end{aligned}$$

■

Now we can show that the two translations preserve equality.

Lemma 9
If t_1 and t_2 are both terms, co-terms, or commands in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ and $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash t_1 = t_2$ then $\lambda\mu_{\text{cons}} \vdash t_1^\sharp = t_2^\sharp$.

Proof
The translation is compositional, so we only need to check each axiom of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$. The η_μ-axiom of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ is precisely given by the η_μ axiom of $\lambda\mu_{\text{cons}}$. The remaining axioms each involve some computation.

- μ:

$$\begin{aligned}\langle\mu\alpha.c\|E\rangle^\sharp &\triangleq E^\sharp[\mu\alpha.c^\sharp]\\ &= [E^-]\mu\alpha.c^\sharp && \text{Lemma 5}\\ &=_\mu c^\sharp[E^-/\alpha]\\ &= c[E/\alpha]^\sharp && \text{Lemma 7}\end{aligned}$$

- $\tilde{\mu}$:

$$\begin{aligned}\langle v\|\tilde{\mu}x.c\rangle^\sharp &\triangleq [\delta]((\lambda x.\mu\delta.c^\sharp)v^\sharp)\\ &= [\delta]\mu\delta.c^\sharp[v^\sharp/x] && \text{Lemma 8}\\ &=_\mu c^\sharp[v^\sharp/x]\\ &= c[v/x]^\sharp && \text{Lemma 7}\end{aligned}$$

- $\eta_{\tilde{\mu}}$:

$$\begin{aligned}(\tilde{\mu}x.\langle x\|e\rangle)^\sharp &\triangleq [\delta]((\lambda x.\mu\delta.e^\sharp[x])\,\Box)\\ &= [\delta](\mu\delta.e^\sharp[\Box]) && \text{Lemma 8}\\ &=_\mu e^\sharp[\Box]\end{aligned}$$

- `surj` :

$$\begin{aligned}(\texttt{car}(E)\cdot\texttt{cdr}(E))^\sharp &= [(\texttt{car}(E)\cdot\texttt{cdr}(E))^-]\Box && \text{Lemma 5}\\ &\triangleq [\texttt{car}\,E^- :: \texttt{cdr}\,E^-]\Box\\ &=_{\eta_{::}} [E^-]\Box\\ &= E^\sharp && \text{Lemma 5}\end{aligned}$$

- `car` :

$$\begin{aligned}(\texttt{car}(v\cdot E))^\sharp &\triangleq \mu\alpha.(v\cdot E)^\sharp[\mu\beta.[\alpha]\texttt{car}\,\beta]\\ &= \mu\alpha.[v^\sharp :: E^-]\mu\beta.[\alpha]\texttt{car}\,\beta && \text{Lemma 5}\\ &=_\mu \mu\alpha.[\alpha]\texttt{car}\,(v^\sharp :: E^-)\\ &=_{\eta_\mu} \texttt{car}\,(v^\sharp :: E^-)\\ &=_{\text{car}} v^\sharp\end{aligned}$$

- `cdr` :

$$\begin{aligned}(\texttt{cdr}(v\cdot E))^\sharp &= [\texttt{cdr}(v\cdot E)]^-\Box && \text{Lemma 5}\\ &\triangleq [\texttt{cdr}\,(v^\sharp :: E^-)]\Box\\ &=_{\text{cdr}} [E^-]\Box\\ &= E^\sharp && \text{Lemma 5}\end{aligned}$$

- exp :

$$\begin{aligned}(\mu[(x\cdot\alpha).c])^\sharp &\triangleq \lambda x.\mu\alpha.c^\sharp\\ &=_{\text{exp}} \mu\beta.[\mathtt{cdr}\ \beta]\mu\alpha.c^\sharp[(\mathtt{car}\ \beta)/x]\\ &=_\mu \mu\beta.c^\sharp[(\mathtt{car}\ \beta)/x,(\mathtt{cdr}\ \beta)/\alpha]\\ &\triangleq \mu\beta.c^\sharp[(\mathtt{car}(\beta))^\sharp/x,(\mathtt{cdr}(\beta))^-/\alpha]\\ &= \mu\beta.(c[\mathtt{car}(\beta)/x,\mathtt{cdr}(\beta)/\alpha])^\sharp && \text{Lemma 7}\\ &\triangleq (\mu\beta.c[\mathtt{car}(\beta)/x,\mathtt{cdr}(\beta)/\alpha])^\sharp\end{aligned}$$

- ς.

$$\begin{aligned}(v\cdot e)^\sharp &\triangleq e^\sharp[\Box\ v^\sharp]\\ &=_{\eta_\mu} e^\sharp[\mu\alpha.[\alpha](\Box\ v^\sharp)]\\ &=_\mu [\delta]\mu\delta.e^\sharp[\mu\alpha.[\alpha](\Box\ v^\sharp)]\\ &= [\delta]((\lambda x.\mu\delta.e^\sharp[\mu\alpha.[\alpha](x\ v^\sharp)])\Box) && \text{Lemma 8}\\ &\triangleq (\tilde\mu x.\langle\mu\alpha.\langle x\|v\cdot\alpha\rangle\|e\rangle)^\sharp\end{aligned}$$

- $\varsigma_{\mathtt{cdr}}$

$$\begin{aligned}\mathtt{cdr}(e)^\sharp &\triangleq e^\sharp[(\mu\beta.[\mathtt{cdr}\ \beta]\Box)]\\ &=_{\eta_\mu} e^\sharp[\mu\alpha.[\alpha](\mu\beta.[\mathtt{cdr}\ \beta]\Box)]\\ &=_\mu [\delta](\mu\delta.e^\sharp[\mu\alpha.[\alpha](\mu\beta.[\mathtt{cdr}\ \beta]\Box)])\\ &= [\delta]((\lambda x.\mu\delta.e^\sharp[\mu\alpha.[\alpha](\mu\beta.[\mathtt{cdr}\ \beta]x)])\Box) && \text{Lemma 8}\\ &\triangleq (\tilde\mu x.\langle\mu\alpha.\langle x\|\mathtt{cdr}(\alpha)\rangle\|e\rangle)^\sharp\end{aligned}$$

- $\varsigma_{\mathtt{car}}$

$$\begin{aligned}\mathtt{car}(e)^\sharp &\triangleq \mu\alpha.e^\sharp[\mu\beta.[\alpha]\mathtt{car}\ \beta]\\ &=_\mu \mu\alpha.e^\sharp[\mu\beta.[\beta](\mu\delta.[\alpha]\mathtt{car}\ \delta)]\\ &=_\mu \mu\alpha.e^\sharp[\mu\beta.[\alpha]\mu\gamma.[\beta](\mu\delta.[\gamma]\mathtt{car}\ \delta)]\\ &\triangleq (\mu\alpha.\langle\mu\beta.\langle\mathtt{car}(\beta)\|\alpha\rangle\|e\rangle)^\sharp\end{aligned}$$

■

Lemma 10
If $\mu\tilde\mu^{\rightarrow}_{\text{cons}} \vdash E_1 = E_2$ then $\lambda\mu_{\text{cons}} \vdash E_1^- = E_2^-$

Proof
We first show the general property that for any streams S, S' and (possibly empty) sequence of terms $v_1, v_2, \ldots, v_{n-1}, v_n$ we have

$$(\lambda\mu_{\text{cons}} \vdash [S](x\ v_1\ v_2\ \ldots\ v_{n-1}\ v_n) = [S']x) \Rightarrow (\lambda\mu_{\text{cons}} \vdash (v_n :: v_{n-1} :: \ldots :: v_2 :: v_1 :: S) = S').$$

This follows by induction on the length of the derivation of $[S](x\ v_1\ v_2\ \ldots\ v_{n-1}\ v_n) = [S']x$. Specifically, the only axioms which can equate a command to $[S](x\ v_1\ v_2\ \ldots\ v_{n-1}\ v_n)$ would be

- an equality internal to S or one of the v_i which carries over to $v_n :: v_{n-1} :: \ldots :: v_2 :: v_1 :: S$ or
- the assoc axiom (in either direction)

$$[v :: S](x\, v_1\, v_2\, \ldots\, v_{n-1}\, v_n) = [S](x\, v_1\, v_2\, \ldots\, v_{n-1}\, v_n\, v)$$

  which directly corresponds to the inductive hypothesis.

Now, Suppose $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash E_1 = E_2$, by Lemma 9 we know that $\lambda\mu_{\text{cons}} \vdash E_1^{\sharp} = E_2^{\sharp}$. By Lemma 5 we further know that $\lambda\mu_{\text{cons}} \vdash E_1^{\sharp} = [E_1^{-}]\Box$ and that $\lambda\mu_{\text{cons}} \vdash E_2^{\sharp} = [E_2^{-}]\Box$. Thus $\lambda\mu_{\text{cons}} \vdash [E_1^{-}]\Box = [E_2^{-}]\Box$ and so for any term v, $\lambda\mu_{\text{cons}} \vdash [E_1^{-}]v = [E_2^{-}]v$. Specifically, $\lambda\mu_{\text{cons}} \vdash [E_1^{-}]x = [E_2^{-}]x$ which by the general property above means that $\lambda\mu_{\text{cons}} \vdash E_1^{-} = E_2^{-}$. ■

Lemma 11

If t_1 and t_2 are terms, streams, or commands in $\lambda\mu_{\text{cons}}$ and $\lambda\mu_{\text{cons}} \vdash t_1 = t_2$ then $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash t_1^{\flat} = t_2^{\flat}$

Proof
The translation is compositional so we only need to check each axiom of $\lambda\mu_{\text{cons}}$. The μ-axiom of $\lambda\mu_{\text{cons}}$ translates to the μ-axiom of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ after Lemma 6. The β_T and assoc axioms each also correspond to single uses of the μ-axiom of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$. The car, cdr and η_μ of $\lambda\mu_{\text{cons}}$ correspond exactly to the axioms of the same names in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ and the $\eta_{::}$ axiom is implemented by `surj`. That leaves two axioms which require some computation:

- exp:

$$\begin{aligned}
(\lambda x.v)^{\flat} &\triangleq \mu[(x\cdot\beta).\langle v^{\flat}\|\beta\rangle] \\
&=_{\text{exp}} \mu\alpha.\langle v^{\flat}[\mathtt{car}(\alpha)/x\|\mathtt{cdr}(\alpha)\rangle \\
&\triangleq \mu\alpha.\langle v^{\flat}[(\text{car }\alpha)^{\flat}/x\|\mathtt{cdr}(\alpha)\rangle \\
&= \mu\alpha.\langle v[(\text{car }\alpha)/x]^{\flat}\|\mathtt{cdr}(\alpha)\rangle && \text{Lemma 6} \\
&\triangleq (\mu\alpha.[\mathtt{cdr}\ \alpha]v[(\text{car }\alpha)/x])^{\flat}
\end{aligned}$$

- $\eta'_{::}$

$$\begin{aligned}
([\mathtt{cdr}\ S](v\ (\mathtt{car}\ S)))^{\flat} &\triangleq \langle\mu\alpha.\langle v^{\flat}\|\mathtt{car}(S^{\flat})\cdot\alpha\rangle\|\mathtt{cdr}S^{\flat}\rangle \\
&=_{\mu} \langle v^{\flat}\|\mathtt{car}(S^{\flat})\cdot\mathtt{cdr}(S^{\flat})\rangle \\
&=_{\mathtt{surj}} \langle v^{\flat}\|S^{\flat}\rangle \\
&\triangleq ([S]v)^{\flat}
\end{aligned}$$

■

The axioms of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ were selected based on the needs of a reduction theory: we wanted to avoid spurious loops. However, it is useful to establish some general equations of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$.

Lemma 12
The three lifting rules of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ can be generalized as equations to apply to all co-terms and not just co-values.

1. $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash \texttt{car}(e) = \mu\alpha.\langle\mu\beta.\langle\texttt{car}(\beta)\|\alpha\rangle\|e\rangle$,
2. $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash \texttt{cdr}(e) = \tilde{\mu}x.\langle\mu\alpha.\langle x\|\texttt{cdr}(\alpha)\rangle\|e\rangle$ and
3. $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash v\cdot e = \tilde{\mu}x.\langle\mu\alpha.\langle x\|v\cdot\alpha\rangle\|e\rangle$

Proof
In each case, either e is a co-value or it is not. If it is not a co-value then this is just the ς rule. If it is then the property follows from the μ, $\tilde{\mu}$, $\eta_{\tilde{\mu}}$, and η_μ rules. ■

Finally, we give the two main lemmas demonstrating that translation functions compose to form an equivalence.

Lemma 13
1. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ commands c, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (c^\sharp)^\flat = c$.
2. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ terms v, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (v^\sharp)^\flat = v$.
3. For all $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$ co-terms e and $\lambda\mu_{\text{cons}}$ terms v, $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \vdash (e^\sharp[v])^\flat = \langle v^\flat\|e\rangle$.

Proof
By mutual induction.

- $$\begin{aligned}((\langle v\|e\rangle)^\sharp)^\flat &\triangleq (e^\sharp[v^\sharp])^\flat \\ &= \langle (v^\sharp)^\flat\|e\rangle && \text{Induction hypothesis} \\ &= \langle v\|e\rangle && \text{Induction hypothesis}\end{aligned}$$

- $$\begin{aligned}((x)^\sharp)^\flat &\triangleq x^\flat \\ &\triangleq x\end{aligned}$$

- $$\begin{aligned}((\mu\alpha.c)^\sharp)^\flat &\triangleq (\mu\alpha.(c^\sharp))^\flat \\ &\triangleq \mu\alpha.((c^\sharp)^\flat) \\ &= \mu\alpha.c && \text{Induction hypothesis}\end{aligned}$$

- $$\begin{aligned}((\mu[(x\cdot\alpha).c])^\sharp)^\flat &\triangleq (\lambda x.\mu\alpha.c^\sharp)^\flat \\ &\triangleq \mu[(x\cdot\beta).\langle\mu\alpha.(c^\sharp)^\flat\|\beta\rangle] \\ &= \mu[(x\cdot\beta).\langle\mu\alpha.c\|\beta\rangle] && \text{Induction hypothesis} \\ &=_\mu \mu[(x\cdot\alpha).c]\end{aligned}$$

- $$\begin{aligned}((\text{car}(e))^\sharp)^\flat &\triangleq (\mu\alpha.e^\sharp[\mu\beta.[\alpha]\text{car }\beta])^\flat \\ &\triangleq \mu\alpha.(e^\sharp[\mu\beta.[\alpha]\text{car }\beta])^\flat \\ &= \mu\alpha.\langle(\mu\beta.[\alpha]\text{car }\beta])^\flat\|e\rangle && \text{Induction hypothesis} \\ &\triangleq \mu\alpha.\langle\mu\beta.\langle\text{car}(\beta)\|\alpha\rangle\|e\rangle \\ &= \texttt{car}(e) && \text{Lemma 12.1}\end{aligned}$$

- $$\begin{aligned}((v' \cdot e)^\sharp[v])^\flat &\triangleq (e^\sharp[\Box\ (v')^\sharp])[v]^\flat \\ &\triangleq e^\sharp[v\ (v')^\sharp]^\flat \\ &= \langle (v\ (v')^\sharp)^\flat \| e\rangle && \text{Induction hypothesis} \\ &\triangleq \langle \mu\alpha.\langle v^\flat \| (v')^\sharp \cdot \alpha\rangle \| e\rangle \\ &= \langle \mu\alpha.\langle v^\flat \| v' \cdot \alpha\rangle \| e\rangle && \text{Induction hypothesis} \\ &=_{\tilde{\mu}} \langle v^\flat \| \tilde{\mu}x.\langle \mu\alpha.\langle x \| v' \cdot \alpha\rangle \| e\rangle\rangle \\ &= \langle v^\flat \| v' \cdot e\rangle && \text{Lemma 12.3}\end{aligned}$$

- $$\begin{aligned}((\alpha)^\sharp[v])^\flat &\triangleq ([\alpha]v)^\flat \\ &= \langle v^\flat \| \alpha\rangle && \text{Induction hypothesis}\end{aligned}$$

- $$\begin{aligned}((\mathtt{cdr}(e))^\sharp[v])^\flat &\triangleq (e^\sharp[\mu\alpha.[\mathtt{cdr}\ \alpha]\Box])[v]^\flat \\ &\triangleq (e^\sharp[\mu\alpha.[\mathtt{cdr}\ \alpha]v])^\flat \\ &= \langle \mu\alpha.\langle v^\flat \| \mathtt{cdr}(\alpha)\rangle \| e\rangle && \text{Induction hypothesis} \\ &=_{\tilde{\mu}} \langle v^\flat \| \tilde{\mu}x.\langle \mu\alpha.\langle x \| \mathtt{cdr}(\alpha)\rangle \| e\rangle\rangle \\ &= \langle v^\flat \| \mathtt{cdr}(e)\rangle && \text{Lemma 12.2}\end{aligned}$$

- $$\begin{aligned}((\tilde{\mu}x.c)^\sharp[v])^\flat &\triangleq ([\gamma]((\lambda x.\mu\gamma.c^\sharp)\ v))^\flat \\ &\triangleq \langle \mu\beta.\langle \mu[(x\cdot\alpha).\langle \mu\gamma.(c^\sharp)^\flat)\|\alpha\rangle] \| v^\flat \cdot \beta\rangle \| \gamma\rangle \\ &= \langle \mu\beta.\langle \mu[(x\cdot\alpha).\langle \mu\gamma.c)\|\alpha\rangle] \| v^\flat \cdot \beta\rangle \| \gamma\rangle && \text{Induction hypothesis} \\ &=_\mu \langle \mu[(x\cdot\alpha).\langle \mu\gamma.c\|\alpha\rangle] \| v^\flat \cdot \gamma\rangle \\ &=_\mu \langle \mu\gamma.c[v^\flat/x] \| \gamma\rangle && \text{Lemma 8} \\ &=_\mu c[v^\flat/x] \\ &=_{\tilde{\mu}} \langle v^\flat \| \tilde{\mu}x.c\rangle\end{aligned}$$

■

Lemma 14

1. For all $\lambda\mu_{\text{cons}}$ commands c, $\lambda\mu_{\text{cons}} \vdash (c^\flat)^\sharp = c$.
2. For all $\lambda\mu_{\text{cons}}$ terms v, $\lambda\mu_{\text{cons}} \vdash (v^\flat)^\sharp = v$.
3. For all $\lambda\mu_{\text{cons}}$ streams S, $\lambda\mu_{\text{cons}} \vdash (S^\flat)^- = S$.

Proof
By mutual induction.

- $$\begin{aligned} (x^\flat)^\sharp &\triangleq x^\sharp \\ &\triangleq x \end{aligned}$$

- $$\begin{aligned} ((\lambda x.v)^\flat)^\sharp &\triangleq (\mu[(x\cdot\alpha).\langle v^\flat \| \alpha\rangle)^\sharp \\ &\triangleq \lambda x.\mu\alpha.[\alpha](v^\flat)^\sharp \\ &= \lambda x.\mu\alpha.[\alpha]v && \text{Inductive hypothesis} \\ &=_{\eta_\mu} \lambda x.v \end{aligned}$$

- $$\begin{aligned} ((v_1\ v_2)^\flat)^\sharp &\triangleq (\mu\alpha.\langle v_1^\flat \| v_2^\flat\cdot\alpha\rangle)^\sharp \\ &\triangleq \mu\alpha.(v_2^\flat\cdot\alpha)^\sharp[(v_1^\flat)^\sharp] \\ &= \mu\alpha.(v_2^\flat\cdot\alpha)^\sharp[v_1] && \text{Inductive hypothesis} \\ &= \mu\alpha.[(v_2^\flat)^\sharp :: \alpha]v_1 && \text{Lemma 5} \\ &= \mu\alpha.[v_2\cdot\alpha]v_1 && \text{Inductive hypothesis} \\ &=_{\text{assoc}} \mu\alpha.[\alpha](v_1\ v_2) \\ &=_{\eta_\mu} v_1\ v_2 \end{aligned}$$

- $$\begin{aligned} ((\mu\alpha.c)^\flat)^\sharp &\triangleq (\mu\alpha.c^\flat)^\sharp \\ &\triangleq \mu\alpha.(c^\flat)^\sharp \\ &= \mu\alpha.c && \text{Inductive hypothesis} \end{aligned}$$

- $$\begin{aligned} ((\text{car } S)^\flat)^\sharp &\triangleq \text{car}(S^\flat)^\sharp \\ &\triangleq \mu\alpha.(S^\flat)^\sharp[\mu\beta.[\alpha]\text{car } \beta] \\ &= \mu\alpha.[(S^\flat)^-]\mu\beta.[\alpha]\text{car } \beta && \text{Lemma 5} \\ &= \mu\alpha.[S]\mu\beta.[\alpha]\text{car } \beta && \text{Inductive hypothesis} \\ &=_\mu \mu\alpha.[\alpha]\text{car } S \\ &=_{\eta_\mu} \text{car } S \end{aligned}$$

- $$\begin{aligned} (([S]v)^\flat)^\sharp &\triangleq (\langle v^\flat \| S^\flat\rangle)^\sharp \\ &\triangleq (S^\flat)^\sharp[(v^\flat)^\sharp] \\ &= [(S^\flat)^-](v^\flat)^\sharp && \text{Lemma 5} \\ &= [S]v && \text{Inductive hypothesis} \end{aligned}$$

- $$\begin{aligned} ((\alpha)^\flat)^- &\triangleq \alpha^- \\ &\triangleq \alpha \end{aligned}$$

- $$\begin{aligned} ((v :: S)^\flat)^- &\triangleq (v^\flat \cdot S^\flat)^- \\ &\triangleq (v^\flat)^\sharp :: (S^\flat)^- \\ &= v :: S && \text{Inductive hypothesis} \end{aligned}$$

- $$\begin{aligned} ((\mathrm{cdr}\ S)^\flat)^- &\triangleq \mathrm{cdr}(S^\flat)^- \\ &\triangleq \mathrm{cdr}\ (S^\flat)^- \\ &= \mathrm{cdr}\ S && \text{Inductive hypothesis} \end{aligned}$$

■

B Equational Correspondence Between $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and $\mu\tilde{\mu}^{\rightarrow}_{\mathbf{cons}}$

We now give details of the proof of the equational correspondence between $\mu\tilde{\mu}^{\rightarrow}_{\eta}$ and $\mu\tilde{\mu}^{\rightarrow}_{\mathrm{cons}}$. We first prove some useful properties about the translations.

The following fact about the macro definitions of cdr and car is useful and is a consequence of simple substitutions:

Fact 7
- $\mu\tilde{\mu}^{\rightarrow}_{\eta} \vdash \mathtt{cdr}(v \cdot E) = E$, and
- $\mu\tilde{\mu}^{\rightarrow}_{\eta} \vdash \mathtt{car}(v \cdot E) = v$.

Proof
By calculation:

- $$\begin{aligned} \mathtt{cdr}(v \cdot E) &\triangleq \tilde{\mu}x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha\rangle] \| v \cdot E\rangle \\ &= \tilde{\mu}x.\langle v \| \tilde{\mu}\_.\langle \mu\alpha.\langle x\|\alpha\rangle \| E\rangle\rangle && \beta \\ &= \langle \mu\alpha.\langle x\|\alpha\rangle \| E\rangle && \tilde{\mu} \\ &= \tilde{\mu}x.\langle x \| E\rangle && \mu \\ &= E && \eta_{\tilde{\mu}} \end{aligned}$$

- $$\begin{aligned} \mathtt{car}(v \cdot E) &\triangleq \mu\alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha\rangle] \| v \cdot\rangle \\ &= \mu\alpha.\langle v \| \tilde{\mu}x.\langle E \| \mu\_.\langle x\|\alpha\rangle\rangle\rangle && \beta \\ &= \mu\alpha.\langle E \| \mu\_.\langle v\|\alpha\rangle\rangle && \tilde{\mu} \\ &= \mu\alpha.\langle v \| \alpha\rangle && \mu \\ &= v && \eta_{\mu} \end{aligned}$$

■

We can now show the main lemma: $\mathtt{cdr}(E)$ is a semantic co-value in $\mu\tilde{\mu}_{\eta}^{\rightarrow}$.

Lemma 15
For all $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ co-values E, $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash \langle \mu\beta.c \| \mathtt{cdr}(E)\rangle = c[\mathtt{cdr}(E)/\beta]$

Proof
It is enough to show that $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash \langle \mu\beta.c \| \mathtt{cdr}(\alpha)\rangle = c[\mathtt{cdr}(\alpha)/\beta]$ since then

$$\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash \langle \mu\beta.c \| \mathtt{cdr}(E)\rangle =_{\mu} \langle \mu\alpha.\langle \mu\beta.c \| \mathtt{cdr}(\alpha)\rangle \| E\rangle = \langle \mu\alpha.c[\mathtt{cdr}(\alpha)/\beta] \| E\rangle =_{\mu} c[\mathtt{cdr}(E)/\beta]$$

$$\begin{aligned}
\langle \mu\beta.c \| \mathtt{cdr}(\alpha)\rangle &= \langle \mu\beta.c \| \tilde{\mu}z.\langle \mu[(x\cdot\gamma).\langle z\|\gamma\rangle] \| \alpha\rangle\rangle & \\
&= \langle \mu[(x\cdot\gamma).\langle \mu\beta.c\|\gamma\rangle] \| \alpha\rangle & \tilde{\mu}\\
&= \langle \mu[(x\cdot\gamma).c[\gamma/\beta] \| \alpha\rangle & \mu\\
&= \langle \mu[(x\cdot\gamma).c[\mathtt{cdr}(x\cdot\gamma)/\beta] \| \alpha\rangle & \text{Fact 7}\\
&= \langle \mu[(x\cdot\gamma).\langle \mu\varepsilon.c[\mathtt{cdr}(\varepsilon)/\beta] \| x\cdot\gamma\rangle \| \alpha\rangle & \mu\\
&= \langle \mu\varepsilon.[\mathtt{cdr}(\varepsilon)/\beta] \| \alpha\rangle & \eta\\
&= c[\mathtt{cdr}(\alpha)/\beta] & \mu
\end{aligned}$$

■

Lemma 16 (*Co-Values remain substitutable*)
For any co-value E of $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$, we have $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash \langle \mu\beta.c \| E^{\circ}\rangle = c[E^{\circ}/\beta]$.

Proof
By induction on E. The case of α is trivial; that leaves

- $$\begin{aligned}
\langle \mu\beta.c \| (v\cdot E)^{\circ}\rangle &\triangleq \langle \mu\beta.c \| v^{\circ}\cdot E^{\circ}\rangle & \\
&= \langle \mu\alpha.\langle \mu\beta.c \| v^{\circ}\cdot\alpha\rangle \| E^{\circ}\rangle & \text{Inductive Hypothesis}\\
&= \langle \mu\alpha.c[(v^{\circ}\cdot\alpha)/\beta \| E^{\circ}\rangle & \mu\\
&= c[(v^{\circ}\cdot E^{\circ})/\beta] & \text{Inductive Hypothesis}
\end{aligned}$$

- $$\begin{aligned}
\langle \mu\beta.c \| \mathrm{cdr}(E)^{\circ}\rangle &\triangleq \langle \mu\beta.c \| \mathrm{cdr}(E^{\circ})\rangle & \\
&= \langle \mu\alpha.\langle \mu\beta.c \| \mathrm{cdr}(\alpha)\rangle \| E^{\circ}\rangle & \text{Inductive Hypothesis}\\
&= \langle \mu\alpha.c[\mathrm{cdr}(\alpha)/\beta] \| E^{\circ}\rangle & \text{Lemma 15}\\
&= c[\mathrm{cdr}(E^{\circ})/\beta] & \text{Inductive Hypothesis}
\end{aligned}$$

■

We now turn to the main result, fleshing out the proof sketched of Theorem 2 that $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ and $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ are in equational correspondence.

Proof

1. If $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash t = t'$ then $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow} \vdash t = t'$, where t and t' range over commands, terms and contexts. We only need to check the η- and β-axioms since all other $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ axioms are $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ axioms. For η we have:

$$\begin{aligned} \mu[(x\cdot\alpha).\langle v\|x\cdot\alpha\rangle] &=_{\texttt{exp}} \mu\beta.\langle v\|\texttt{car}(\beta)\cdot\texttt{cdr}(\beta)\rangle \\ &=_{\texttt{surj}} \mu\beta.\langle v\|\beta\rangle \\ &=_{\eta_\mu} v \end{aligned}$$

The derivability of β follows from the exp rule together with projection operations, the underlying substitution calculus, and the derivability of the unrestricted version of the lifting rules.

$$\begin{aligned} \langle\mu[(x\cdot\alpha).c]\|v\cdot e\rangle &= \langle\mu\beta.c[\text{car}(\beta)/x,\text{cdr}(\beta)/\alpha]\|v\cdot e\rangle && \text{exp} \\ &= \langle\mu\beta.c[\text{car}(\beta)/x,\text{cdr}(\beta)/\alpha]\|\tilde{\mu}y.\langle\mu\gamma.\langle y\|v\cdot\gamma\rangle\|e\rangle\rangle && \varsigma. \\ &= \langle\mu\gamma.\langle\mu\beta.c[\text{car}(\beta)/x,\text{cdr}(\beta)/\alpha]\|v\cdot\gamma\rangle\|e\rangle && \tilde{\mu} \\ &= \langle\mu\gamma.c[\text{car}(v\cdot\gamma)/x,\text{cdr}(v\cdot\gamma)/\alpha]\|e\rangle && \mu \\ &= \langle\mu\gamma.c[v/x,\gamma/\alpha]\|e\rangle && \texttt{car/cdr} \\ &= \langle\mu\alpha.c[v/x]\|e\rangle && \alpha \\ &= \langle v\|\tilde{\mu}x.\langle\mu\alpha.c\|e\rangle\rangle && \tilde{\mu} \end{aligned}$$

2. If $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow} \vdash t = t'$ then $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash t^\circ = (t')^\circ$. This follows by checking the axioms. The η_μ- and $\eta_{\tilde{\mu}}$-axioms come directly from the equivalent axioms of $\mu\tilde{\mu}_{\eta}^{\rightarrow}$. The $\tilde{\mu}$-axiom works using the $\tilde{\mu}$-axiom of $\mu\tilde{\mu}_{\eta}^{\rightarrow}$ and the compositionality of the $^\circ$ translation that gives $c^\circ[v^\circ/x] \triangleq c[v/x]^\circ$. The μ-axiom works since by Lemma 16 $\langle\mu\alpha.c^\circ\|E^\circ\rangle = c^\circ[E^\circ/\alpha]$ and because the translation is compositional that further equals $c[E/\alpha]^\circ$.

- $\mu\tilde{\mu}_{\eta}^{\rightarrow} \vdash (\texttt{car}(E)\cdot\texttt{cdr}(E))^\circ = E^\circ$

$$\begin{aligned} &(\texttt{car}(E)\cdot\texttt{cdr}(E))^\circ \\ &\triangleq \texttt{car}(E^\circ)\cdot\texttt{cdr}(E^\circ) \\ &= \tilde{\mu}f.\langle f\|\texttt{car}(E^\circ)\cdot\texttt{cdr}(E^\circ)\rangle && \eta_{\tilde{\mu}} \\ &= \tilde{\mu}f.\langle\mu\alpha.\langle f\|\texttt{car}(\alpha)\cdot\texttt{cdr}(\alpha)\rangle\|E^\circ\rangle && \text{Lemma 16} \\ &= \tilde{\mu}f.\langle\mu[(x\cdot\beta).\langle\mu\alpha.\langle f\|\texttt{car}(\alpha)\cdot\texttt{cdr}(\alpha)\rangle\|x\cdot\beta\rangle]\|E^\circ\rangle && \eta \\ &= \tilde{\mu}f.\langle\mu[(x\cdot\beta).\langle f\|\texttt{car}(x\cdot\beta)\cdot\texttt{cdr}(x\cdot\beta)\rangle]\|E^\circ\rangle && \mu \\ &= \tilde{\mu}f.\langle\mu[(x\cdot\beta).\langle f\|x\cdot\texttt{cdr}(x\cdot\beta)\rangle]\|E^\circ\rangle && \text{Fact 7} \\ &= \tilde{\mu}f.\langle\mu[(x\cdot\beta).\langle f\|x\cdot\beta\rangle]\|E^\circ\rangle && \text{Fact 7} \\ &= \tilde{\mu}f.\langle f\|E^\circ\rangle && \eta \\ &= E^\circ && \eta_{\tilde{\mu}} \end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash (\mu[(x\cdot\alpha).c])^\circ = (\mu\beta.c[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha])^\circ$

$$\begin{aligned}
\mu[(x\cdot\alpha).c]^\circ &\triangleq \mu[(x\cdot\alpha).c^\circ] \\
&= \mu\beta.\langle\mu[(x\cdot\alpha).c^\circ]\|\beta\rangle && \eta_\mu \\
&= \mu\beta.\langle\mu[(x\cdot\alpha).c^\circ]\|\mathtt{car}(\beta)\cdot\mathtt{cdr}(\beta)\rangle && \mathtt{surj}\text{ Shown above} \\
&= \mu\beta.\langle\mathtt{car}(\beta)\|\tilde{\mu}x.\langle\mu\alpha.c^\circ\|\mathtt{cdr}(\beta)\rangle\rangle && \beta \\
&= \mu\beta.\langle\mu\alpha.c^\circ[\mathtt{car}(\beta)/x]\|\mathtt{cdr}(\beta)\rangle && \tilde{\mu} \\
&= \mu\beta.c[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha] && \text{Lemma 15} \\
&\triangleq (\mu\beta.c[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha])^\circ
\end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash \mathtt{car}(v\cdot E)^\circ = v^\circ$

$$\begin{aligned}
\mathtt{car}(v\cdot E)^\circ &\triangleq \mu\alpha.\langle\mu[(x\cdot\_).\langle x\|\alpha\rangle]\|v^\circ\cdot E^\circ\rangle \\
&= \mu\alpha.\langle v^\circ\|\tilde{\mu}x.\langle E^\circ\|\mu\_.\langle x\|\alpha\rangle\rangle\rangle && \beta \\
&= \mu\alpha.\langle E^\circ\|\mu\_.\langle v^\circ\|\alpha\rangle\rangle && \tilde{\mu} \\
&= \mu\alpha.\langle v^\circ\|\alpha\rangle && \text{Lemma 16} \\
&= v^\circ && \eta_\mu
\end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash \mathtt{cdr}(v\cdot E)^\circ = E^\circ$

$$\begin{aligned}
\mathtt{cdr}(v\cdot E)^\circ &\triangleq \tilde{\mu}x.\langle\mu[(\_\cdot\alpha).\langle x\|\alpha\rangle]\|v^\circ\cdot E^\circ\rangle \\
&= \tilde{\mu}x.\langle v^\circ\|\tilde{\mu}\_.\langle\mu\alpha.\langle x\|\alpha\rangle\|E^\circ\rangle\rangle && \beta \\
&= \langle\mu\alpha.\langle x\|\alpha\rangle\|E^\circ\rangle && \tilde{\mu} \\
&= \tilde{\mu}x.\langle x\|E^\circ\rangle && \text{Lemma 16} \\
&= E^\circ && \eta_{\tilde{\mu}}
\end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash \mathtt{car}(e)^\circ = (\mu\alpha.\langle\mu\beta.\langle\mathtt{car}(\beta)\|\alpha\rangle\|e\rangle)^\circ$

$$\begin{aligned}
\mathtt{car}(e) &\triangleq \mu\alpha.\langle\mu[(x\cdot\_).\langle x\|\alpha\rangle]\|e^\circ\rangle \\
&= \mu\alpha.\langle\mu\beta.\langle\mu[(x\cdot\_).\langle x\|\alpha\rangle]\|\beta\rangle\|e^\circ\rangle && \eta_\mu \\
&= \mu\alpha.\langle\mu\beta.\langle\mu\gamma.\langle\mu[(x\cdot\_).\langle x\|\gamma\rangle]\|\beta\rangle\|\alpha\rangle\|e^\circ\rangle && \mu \\
&\triangleq (\mu\alpha.\langle\mu\beta.\langle\mathtt{car}(\beta)\|\alpha\rangle\|e\rangle)^\circ
\end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash (\mathtt{cdr}(e))^\circ = (\tilde{\mu}x.\langle\mu\alpha.\langle x\|\mathtt{cdr}(\alpha)\rangle\|e\rangle)^\circ$

$$\begin{aligned}
(\mathtt{cdr}(e))^\circ &\triangleq \tilde{\mu}x.\langle\mu[(\_\cdot\beta).\langle x\|\beta\rangle]\|e^\circ\rangle \\
&= \tilde{\mu}x.\langle\mu\alpha.\langle\mu[(\_\cdot\beta).\langle x\|\beta\rangle]\|\alpha\rangle\|e^\circ\rangle && \eta_\mu \\
&= \tilde{\mu}x.\langle\mu\alpha.\langle x\|\tilde{\mu}y.\langle\mu[(\_\cdot\beta).\langle y\|\beta\rangle]\|\alpha\rangle\rangle\|e^\circ\rangle && \tilde{\mu} \\
&\triangleq (\tilde{\mu}x.\langle\mu\alpha.\langle x\|\mathtt{cdr}(\alpha)\rangle\|e\rangle)^\circ
\end{aligned}$$

- $\mu\tilde{\mu}_\eta^{\rightarrow} \vdash (v \cdot e)^\circ = (\tilde{\mu}x.\langle \mu\alpha.\langle x \| v \cdot \alpha\rangle \| e\rangle)^\circ$

$$\begin{aligned}
(v \cdot e)^\circ &\triangleq v^\circ \cdot e^\circ \\
&= \tilde{\mu}x.\langle x \| v^\circ \cdot e^\circ\rangle && \eta_{\tilde{\mu}} \\
&= \tilde{\mu}x.\langle \mu[(y \cdot \alpha).\langle x \| y \cdot \alpha\rangle] \| v^\circ \cdot e^\circ\rangle && \eta \\
&= \tilde{\mu}x.\langle v^\circ \| \tilde{\mu}y.\langle \mu\alpha.\langle x \| y \cdot \alpha\rangle \| e^\circ\rangle\rangle && \beta \\
&= \tilde{\mu}x.\langle \mu\alpha.\langle x \| v^\circ \cdot \alpha\rangle \| e^\circ\rangle && \tilde{\mu} \\
&= (\tilde{\mu}x.\langle \mu\alpha.\langle x \| v \cdot \alpha\rangle \| e\rangle)^\circ
\end{aligned}$$

3. If t is in $\mu\tilde{\mu}_\eta^{\rightarrow}$ then $t^\circ \triangleq t$. That is, the $^\circ$ translation is the identity on everything except $\mathtt{car}(e)$ and $\mathtt{cdr}(e)$ as constants which do not appear in the $\mu\tilde{\mu}_\eta^{\rightarrow}$ sub-syntax.
4. If t is in $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$ then $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow} \vdash t^\circ = t$. By induction on t. The only non-trivial cases to handle are $\mathtt{car}(e)$ and $\mathtt{cdr}(e)$. If e is a co-value, we have:

$$\begin{aligned}
\mathtt{car}(E)^\circ &\triangleq \mu\alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha\rangle] \| E^\circ\rangle \\
&= \mu\alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha\rangle] \| E\rangle && \text{Inductive hypothesis} \\
&= \mu\alpha.\langle \mu\beta.\langle \mathtt{car}(\beta) \| \alpha\rangle \| E\rangle && \text{exp} \\
&= \mu\alpha.\langle \mathtt{car}(E) \| \alpha\rangle && \mu \\
&= \mathtt{car}(E) && \eta_\mu \\
\mathtt{cdr}(E)^\circ &\triangleq \tilde{\mu}x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha\rangle] \| E^\circ\rangle \\
&= \tilde{\mu}x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha\rangle] \| E\rangle && \text{Inductive hypothesis} \\
&= \tilde{\mu}x.\langle \mu\beta.\langle x \| \mathtt{cdr}(\beta)\rangle \| E\rangle && \text{exp} \\
&= \tilde{\mu}x.\langle x \| \mathtt{cdr}(E)\rangle && \mu \\
&= \mathtt{cdr}(E) && \eta_{\tilde{\mu}}
\end{aligned}$$

In the case where e is not a co-value:

$$\begin{aligned}
\mathtt{car}(e)^\circ &\triangleq \mu\alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha\rangle] \| e^\circ\rangle \\
&= \mu\alpha.\langle \mu[(x \cdot \_).\langle x \| \alpha\rangle] \| e\rangle && \text{Inductive hypothesis} \\
&= \mu\alpha.\langle \mu\beta.\langle \mathtt{car}(\beta) \| \alpha\rangle \| e\rangle && \text{exp} \\
&= \mathtt{car}(e) && \varsigma_{\text{car}} \\
\mathtt{cdr}(e)^\circ &\triangleq \tilde{\mu}x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha\rangle] \| e^\circ\rangle \\
&= \tilde{\mu}x.\langle \mu[(\_ \cdot \alpha).\langle x \| \alpha\rangle] \| e\rangle && \text{Inductive hypothesis} \\
&= \tilde{\mu}x.\langle \mu\beta.\langle x \| \mathtt{cdr}(\beta)\rangle \| e\rangle && \text{exp} \\
&= \mathtt{cdr}(e) && \varsigma_{\text{cdr}}
\end{aligned}$$

■

C A stack calculus and its reflection

In this section, we further explicate the details of the reflection of the stack calculus in $\mu\tilde{\mu}_{\text{cons}}^{\rightarrow}$, showing first the details of the reflection of the focalized syntax in the full

$$\begin{aligned}
x^{\varsigma} &\triangleq x \\
\alpha^{\varsigma} &\triangleq \alpha \\
(\mu\alpha.c)^{\varsigma} &\triangleq \mu\alpha.(c^{\varsigma}) \\
\mu[(x\cdot\alpha).c]^{\varsigma} &\triangleq \mu[(x\cdot\alpha).c^{\varsigma}] \\
(\tilde{\mu}x.c)^{\varsigma} &\triangleq \tilde{\mu}x.(c^{\varsigma}) \\
\mathtt{car}(E)^{\varsigma} &\triangleq \mathtt{car}(E^{\varsigma}) \\
\mathtt{car}(e)^{\varsigma} &\triangleq \mu\alpha.\langle\mu\beta.\langle\mathtt{car}(\beta)\|\alpha\rangle\|e^{\varsigma}\rangle && (e \notin \text{Co-Values}) \\
\mathtt{cdr}(E)^{\varsigma} &\triangleq \mathtt{cdr}(E^{\varsigma}) \\
\mathtt{cdr}(e)^{\varsigma} &\triangleq \tilde{\mu}x.\langle\mu\alpha.\langle x\|\mathtt{cdr}(\alpha)\rangle\|e^{\varsigma}\rangle && (e \notin \text{Co-Values}) \\
(v\cdot E)^{\varsigma} &\triangleq v^{\varsigma}\cdot E^{\varsigma} \\
(v\cdot e)^{\varsigma} &\triangleq \tilde{\mu}x.\langle\mu\alpha.\langle x\|v^{\varsigma}\cdot\alpha\rangle\|e^{\varsigma}\rangle && (e \notin \text{Co-Values}) \\
\langle v\|e\rangle^{\varsigma} &\triangleq \langle v^{\varsigma}\|e^{\varsigma}\rangle
\end{aligned}$$

Fig. C 1. The $(-)^{\varsigma} : \mu\tilde{\mu}^{\rightarrow}_{\text{cons}} \to \mu\tilde{\mu}^{F}_{\text{cons}}$ translation for computing focalized programs

system, and then the reflection of the stack calculus into the smaller focalized intermediate language.

Each reflection arises from normalization with respect to a subset of the rules. As such, in this appendix we present evaluation functions for computing ς and $\tilde{\mu}\mathtt{exp}$-normal forms. Such evaluation functions greatly simplify the proofs of commutation between normalization and the rest of the reduction theory.

Lemma 17 (*Closure of $\mu\tilde{\mu}^{F}_{cons}$*)

1. Closure under substitution: If t, v, E are in the focalized sub-syntax, then $t[v/x, E/\alpha]$ is in the same syntactic category as t in $\mu\tilde{\mu}^{F}_{\text{cons}}$.
2. Closure under reduction: If t is in the focalized sub-syntax and $t \to t'$ according to the rules of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$, then t' is in the focalized sub-syntax (and in the same syntactic category as t).

Proof

The first point follows by induction on t. For the second point, we proceed by cases. The μ, $\tilde{\mu}$, and exp rules hold by the closure under substitution. The η, car, and cdr rules are all immediate. Finally, the various ς-rules are simply syntactically prohibited as there left hand sides are not focalized because of the requirement that the lifted e is not a co-value. ■

The procedure to compute a terms ς-normal form is given in Figure C 1.

Lemma 18

$(-)^{\varsigma}$ computes the unique ς-normal form of any command, term, or co-term in $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$

Proof

Observe that E^{ς} is always a co-value. Therefore, by cases, we know that t^{ς} is in $\mu\tilde{\mu}^{F}_{\text{cons}}$ and that it is a ς-normal form.

By induction we see that $t \twoheadrightarrow_{\varsigma} t^{\varsigma}$. x^{ς} and α^{ς} work in zero steps, $(\mu\alpha.c)^{\varsigma}$, $\mu[(x\cdot\alpha).c]^{\varsigma}$, $(\tilde{\mu}x.c)^{\varsigma}$, $\mathtt{car}(E)^{\varsigma}$, $\mathtt{cdr}(E)^{\varsigma}$, $(v\cdot E)^{\varsigma}$, and $\langle v\|e\rangle^{\varsigma}$ work by the inductive hypothesis, the remaining cases of $\mathtt{car}(e)^{\varsigma}$, $\mathtt{cdr}(e)^{\varsigma}$, $(v \cdot e)^{\varsigma}$ each correspond to a single application of a ς rule followed by additional ς reductions given by the inductive hypothesis.

Since ς is a (linear) term rewriting system without any critical pairs it is confluent and so must have unique normal forms. Thus, t^{ς} is the unique ς-normal form of t. ■

Lemma 19

For any t,v,E,x, and α, $t^{\varsigma}[v^{\varsigma}/x] \triangleq (t[v/x])^{\varsigma}$ and $t^{\varsigma}[E^{\varsigma}/\alpha] \triangleq (t[E/\alpha])^{\varsigma}$.

Proof

Induction on t. The only trick is that substitution will never change what is a co-value. ■

Lemma 20

ς-normalization commutes across the non-ς reductions of $\mu\tilde{\mu}^{\rightarrow}_{\text{cons}}$.

Proof

By Lemma 18 it is enough to show that if $t \twoheadrightarrow t'$ then $t^{\varsigma} \twoheadrightarrow (t')^{\varsigma}$. By inducting over the reduction $t \twoheadrightarrow t'$ it is sufficient to show that $t \rightarrow t'$ implies $t^{\varsigma} \twoheadrightarrow (t')^{\varsigma}$. Examining the cases, the main issue comes when we have an internal reduction into a (co-)term subject to lifting as reduction might turn a non co-value co-term into a co-value and thus change the translation. The main idea then is to prove the special case that if $e \rightarrow E$ then $e^{\varsigma} \twoheadrightarrow E^{\varsigma}$. This follows by induction on the derivation of the reduction $e \rightarrow E$, with only a small number of cases to consider:

- If $\tilde{\mu}x.\langle x \| E\rangle \rightarrow E$

$$\begin{aligned} (\tilde{\mu}x.\langle x \| E\rangle)^{\varsigma} &\triangleq \tilde{\mu}x.\langle x \| E^{\varsigma}\rangle \\ &\rightarrow E^{\varsigma} && \eta_{\tilde{\mu}} \end{aligned}$$

- If $v \cdot e \rightarrow v \cdot E$

$$\begin{aligned} (v \cdot e)^{\varsigma} &\triangleq \tilde{\mu}x.\langle \mu\alpha.\langle x \| v^{\varsigma} \cdot \alpha\rangle \| e^{\varsigma}\rangle \\ &\twoheadrightarrow \tilde{\mu}x.\langle \mu\alpha.\langle x \| v^{\varsigma} \cdot \alpha\rangle \| E^{\varsigma}\rangle && \text{Inductive Hypothesis} \\ &\rightarrow \tilde{\mu}x.\langle x \| v^{\varsigma} \cdot E^{\varsigma}\rangle && \mu \\ &\rightarrow v^{\varsigma} \cdot E^{\varsigma} && \eta_{\tilde{\mu}} \\ &\triangleq (v \cdot E)^{\varsigma} \end{aligned}$$

- If $\mathtt{cdr}(e) \rightarrow \mathtt{cdr}(E)$

$$\begin{aligned} \mathtt{cdr}(e)^{\varsigma} &\triangleq \tilde{\mu}x.\langle \mu\alpha.\langle x \| \mathtt{cdr}(\alpha)\rangle \| e^{\varsigma}\rangle \\ &\twoheadrightarrow \tilde{\mu}x.\langle \mu\alpha.\langle x \| \mathtt{cdr}(\alpha)\rangle \| E^{\varsigma}\rangle && \text{Inductive Hypothesis} \\ &\rightarrow \tilde{\mu}x.\langle x \| \mathtt{cdr}(E^{\varsigma})\rangle && \mu \\ &\rightarrow \mathtt{cdr}(E^{\varsigma}) && \eta_{\tilde{\mu}} \\ &\triangleq \mathtt{cdr}(E)^{\varsigma} \end{aligned}$$

- If $\mathtt{car}(e) \to \mathtt{car}(E)$

$$\begin{aligned}
\mathtt{car}(e)^{\varsigma} &\triangleq \mu\alpha.\langle \mu\beta.\langle \mathtt{car}(\beta)\|\alpha\rangle\|e^{\varsigma}\rangle \\
&\twoheadrightarrow \mu\alpha.\langle \mu\beta.\langle \mathtt{car}(\beta)\|\alpha\rangle\|E^{\varsigma}\rangle && \text{Inductive Hypothesis} \\
&\to \mu\alpha.\langle \mathtt{car}(E^{\varsigma})\|\alpha\rangle && \mu \\
&\to \mathtt{car}(E^{\varsigma}) && \eta_\mu \\
&\triangleq \mathtt{car}(E)^{\varsigma}
\end{aligned}$$

Otherwise, the translation is completely compositional and so the cases are direct:

- If $\langle \mu\alpha.c\|E\rangle \to_\mu c[E/\alpha]$ we have:

$$\begin{aligned}
\langle \mu\alpha.c\|E\rangle^{\varsigma} &\triangleq \langle \mu\alpha.c^{\varsigma}\|E^{\varsigma}\rangle \\
&\to_\mu c^{\varsigma}[E^{\varsigma}/\alpha] \\
&=_\alpha c[E/\alpha]^{\varsigma} && \text{Lemma 19.}
\end{aligned}$$

- If $\langle v\|\tilde{\mu}x.c\rangle \to_{\tilde{\mu}} c[v/x]$ we have:

$$\begin{aligned}
\langle v\|\tilde{\mu}x.c\rangle^{\varsigma} &\triangleq \langle v^{\varsigma}\|\tilde{\mu}x.c^{\varsigma}\rangle \\
&\to_\mu c^{\varsigma}[v^{\varsigma}/x] \\
&=_\alpha c[v/x]^{\varsigma} && \text{Lemma 19.}
\end{aligned}$$

- If $\mu\alpha.\langle v\|\alpha\rangle \to_{\eta_\mu} v$ we have: $(\mu\alpha.\langle v\|\alpha\rangle)^{\varsigma} \triangleq \mu\alpha.\langle v^{\varsigma}\|\alpha\rangle \to_{\eta_\mu} v^{\varsigma}$.
- If $\tilde{\mu}x.\langle x\|e\rangle \to_{\eta_{\tilde{\mu}}} e$ we have: $(\tilde{\mu}x.\langle x\|e\rangle)^{\varsigma} \triangleq \tilde{\mu}x.\langle x\|e^{\varsigma}\rangle \to_{\eta_{\tilde{\mu}}} e^{\varsigma}$.
- If $\mu[(x\cdot\alpha).c] \to_{\mathrm{exp}} \mu\beta.c[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha]$ we have:

$$\begin{aligned}
\mu[(x\cdot\alpha).c]^{\varsigma} &\triangleq \mu[(x\cdot\alpha).c^{\varsigma}] \\
&\to_{\mathrm{exp}} \mu\beta.c^{\varsigma}[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha] \\
&\triangleq \mu\beta.c[\mathtt{car}(\beta)/x, \mathtt{cdr}(\beta)/\alpha]^{\varsigma} && \text{Lemma 19.}
\end{aligned}$$

- If $\mathtt{car}(v\cdot E) \to_{\mathrm{car}} v$ we have: $\mathtt{car}(v^{\varsigma}\cdot E^{\varsigma}) \to_{\mathrm{car}} v^{\varsigma}$.
- If $\mathtt{cdr}(v\cdot E) \to_{\mathrm{cdr}} E$ we have: $\mathtt{cdr}(v^{\varsigma}\cdot E^{\varsigma}) \to_{\mathrm{cdr}} E^{\varsigma}$.
- In the case of $\mathtt{car}(E)\cdot\mathtt{cdr}(E) \to_{\mathtt{surj}} E$ we have $(\mathtt{car}(E)\cdot\mathtt{cdr}(E))^{\varsigma} \triangleq \mathtt{car}(E^{\varsigma})\cdot \mathtt{cdr}(E^{\varsigma}) \to_{\mathtt{surj}} E^{\varsigma}$, since E^{ς} is still a co-value.

■

Lemma 1 follows as the composition of Lemma 20 and Theorem 4.

The function $(-)^{\tilde{\mu}}$ translates commands and terms in the focalized sub-syntax ($\mu\tilde{\mu}^{F}_{\mathrm{cons}}$, Figure 11) into Σ^x and is given in Figure C 2.

Lemma 21 (*Closure of* Σ^x)

1. – Closure under substitution: If t, v, E are in Σ^x then $t[v/x, E/\alpha]$ is in the same syntactic category as t in Σ^x.
2. – Closure under reduction: If t is in Σ^x and $t \to t'$ according to the rules of $\mu\tilde{\mu}^{\to}_{\mathrm{cons}}$ then t' is in Σ^x (in the same syntactic category as t).

Proof

The first point follows by induction on t. The second point by cases. ■

$$\begin{aligned}
\langle v \| \tilde{\mu}x.c\rangle^{\tilde{\mu}} &\triangleq c^{\tilde{\mu}}[v^{\tilde{\mu}}/x] \\
\langle v \| E\rangle^{\tilde{\mu}} &\triangleq \langle v^{\tilde{\mu}} \| E^{\tilde{\mu}}\rangle \\
x^{\tilde{\mu}} &\triangleq x \\
(\mu\alpha.c)^{\tilde{\mu}} &\triangleq \mu\alpha.(c^{\tilde{\mu}}) \\
\mu[(x\cdot\alpha).c]^{\tilde{\mu}} &\triangleq \mu\beta.c^{\tilde{\mu}}[\texttt{car}(\beta)/x, \texttt{cdr}(\beta)/\alpha] \\
\texttt{car}(E)^{\tilde{\mu}} &\triangleq \texttt{car}(E^{\tilde{\mu}}) \\
\alpha^{\tilde{\mu}} &\triangleq \alpha \\
\texttt{cdr}(E)^{\tilde{\mu}} &\triangleq \texttt{cdr}(E^{\tilde{\mu}}) \\
(v\cdot E)^{\tilde{\mu}} &\triangleq v^{\tilde{\mu}}\cdot E^{\tilde{\mu}}
\end{aligned}$$

Fig. C 2. Translation from $\mu\tilde{\mu}^{F}_{\text{cons}}$ to Σ^{x}

Lemma 22

For every t, v, E in the focalized sub-syntax, $t[v/x, E/\alpha]^{\tilde{\mu}} \triangleq t^{\tilde{\mu}}[v^{\tilde{\mu}}/x, E^{\tilde{\mu}}/\alpha]$.

Proof

By induction on t. ■

Lemma 23

$-^{\tilde{\mu}}$ computes unique normal forms of $\mu\tilde{\mu}^{F}_{\text{cons}}$ with respect to the $\tilde{\mu}$- and exp-rules.

Proof

For every t in $\mu\tilde{\mu}^{F}_{\text{cons}}$, $t \twoheadrightarrow t^{\tilde{\mu}}$. This holds by induction on t. In each case we need only to perform at most one $\tilde{\mu}$- or `exp`-reduction and then utilize the inductive hypothesis. $t^{\tilde{\mu}}$ does not contains any $\tilde{\mu}$- or `exp`-redexes and is thus a normal form. $\tilde{\mu}$`exp`-reduction is confluent because it lacks critical pairs. ■

Intermezzo 8

Note that the above Lemma implies that $\tilde{\mu}$`exp`-reduction is normalizing as the $(-)^{\tilde{\mu}}$ function computes a normal form with respect to it. However, it is interesting to think about why $\tilde{\mu}$-reduction is normalizing (the `exp`-rule is clearly strongly normalizing on its own as the number of $\mu[(x\cdot\alpha).c]$ forms is strictly decreasing). The basic idea for how we could go about showing $\tilde{\mu}$-normalization is that $\tilde{\mu}$-reduction never introduces any new redexes. In particular if v is $\tilde{\mu}$-normal then $c[v/x]$ has one fewer $\tilde{\mu}$-redex then $\langle v \| \tilde{\mu}x.c\rangle$. Thus, there is clearly a reduction order which is normalizing. Further, strong normalization could be additionally shown by interpreting each (co-)term or command as a polynomial with positive integer coefficients.

$$\begin{aligned}
[\![\langle v \| E \rangle]\!] &= [\![v]\!] + [\![E]\!] \\
[\![\langle v \| \tilde{\mu} x.c \rangle]\!] &= 1 + [\![v]\!] + [\![c]\!][[\![v]\!]/x] \\
[\![x]\!] &= x \\
[\![\alpha]\!] &= 0 \\
[\![v \cdot E]\!] &= [\![v]\!] + [\![E]\!] \\
[\![\mathrm{cdr}(E)]\!] &= [\![E]\!] \\
[\![\mathrm{car}(E)]\!] &= [\![E]\!] \\
[\![\mu\alpha.c]\!] &= [\![c]\!] \\
[\![\mu[(x\cdot\alpha).c]]\!] &= [\![c]\!][0/x] \\
[\![\tilde{\mu}x.c]\!] &= [\![c]\!][0/x]
\end{aligned}$$

Then, the general property we would show is that $[\![t[v/x]]\!] = [\![t]\!][[\![v]\!]/x]$, and then given any assignment ϕ of free variables in the polynomial to integers if $t \to_{\tilde{\mu}} t'$ then $[\![t]\!](\phi) > [\![t']\!](\phi)$ and so $\tilde{\mu}$ is strongly normalizing.

That $\tilde{\mu}$ and μ are each strongly normalizing on their own is an interesting property of the two-sided sequent calculus, however, as we have given a translation function we do not actually need it here.

Lemma 24
$\tilde{\mu}$exp-normalization commutes across $\mu\tilde{\mu}^F_{\text{cons}}$

Proof
Equivalently, $t \to t'$ by a rule other than $\eta_{\tilde{\mu}}$ implies $t^{\tilde{\mu}} \twoheadrightarrow (t')^{\tilde{\mu}}$. We can prove this by induction on the reduction $t \to t'$. Reductions not at the top of t are nearly all automatic since $-^{\tilde{\mu}}$ is compositional except for the cases of $\langle v \| \mu x.c\rangle$ and $\mu[(x\cdot\alpha).c]$ which are handled by Lemma 22. That leaves only the cases where the reduction happens at the top of t:

- In the case of $\langle \mu\alpha.c \| E\rangle \to_\mu c[E/\alpha]$ we have $\langle \mu\alpha.c \| E\rangle^{\tilde{\mu}} \triangleq \langle \mu\alpha.c^{\tilde{\mu}} \| E^{\tilde{\mu}}\rangle \to c^{\tilde{\mu}}[E^{\tilde{\mu}}/\alpha]$ and by Lemma 22 $c^{\tilde{\mu}}[E^{\tilde{\mu}}/\alpha] =_\alpha c[E/\alpha]^{\tilde{\mu}}$.
- In the case of $\langle v \| \tilde{\mu}x.c\rangle \to_{\tilde{\mu}} c[v/x]$ we have $\langle v \| \tilde{\mu}x.c\rangle^{\tilde{\mu}} \triangleq c^{\tilde{\mu}}[v^{\tilde{\mu}}/x]$ which by Lemma 22 is $c[v/x]^{\tilde{\mu}}$.
- In the case of $\mu\alpha.\langle v\|\alpha\rangle \to_{\eta_\mu} v$ we have $(\mu\alpha.\langle v\|\alpha\rangle)^{\tilde{\mu}} \triangleq \mu\alpha.\langle v^{\tilde{\mu}}\|\alpha\rangle \to_{\eta_\mu} v^{\tilde{\mu}}$.
- In the case of $\langle v\|\tilde{\mu}x.\langle x\|\tilde{\mu}y.c\rangle\rangle \to_{\eta_{\tilde{\mu}}} \langle v\|\tilde{\mu}y.c\rangle$ we have $\langle v\|\tilde{\mu}x.\langle x\|\tilde{\mu}y.c\rangle\rangle^{\tilde{\mu}} \triangleq c^{\tilde{\mu}}[v/y]$ and $\langle v\|\tilde{\mu}y.c\rangle^{\tilde{\mu}} \triangleq c^{\tilde{\mu}}[v/y]$.
- In the case of $\mu[(x\cdot\alpha).c] \to_{\text{exp}} \mu\beta.c[\texttt{car}(\beta)/x, \texttt{cdr}(\beta)/\alpha]$ we have $\mu[(x\cdot\alpha).c]^{\tilde{\mu}} \triangleq \mu\beta.c^{\tilde{\mu}}[\texttt{car}(\beta)/x, \texttt{cdr}(\beta)/\alpha]$ which is $(\mu\beta.c[\texttt{car}(\beta)/x, \texttt{cdr}(\beta)/\alpha])^{\tilde{\mu}}$ by Lemma 22.
- In the case of $\texttt{car}(v\cdot E) \to_{\text{car}} v$ we have $\texttt{car}(v\cdot E)^{\tilde{\mu}} \triangleq \texttt{car}(v^{\tilde{\mu}}\cdot E^{\tilde{\mu}}) \to_{\text{car}} v^{\tilde{\mu}}$.
- In the case of $\texttt{cdr}(v\cdot E) \to_{\text{cdr}} E$ we have $\texttt{cdr}(v\cdot E)^{\tilde{\mu}} \triangleq \texttt{cdr}(v^{\tilde{\mu}}\cdot E^{\tilde{\mu}}) \to_{\text{cdr}} E^{\tilde{\mu}}$.
- In the case of $\texttt{car}(E)\cdot\texttt{cdr}(E) \to_{\texttt{surj}} E$ we have $(\texttt{car}(E)\cdot\texttt{cdr}(E))^{\tilde{\mu}} \triangleq \texttt{car}(E^{\tilde{\mu}})\cdot\texttt{cdr}(E^{\tilde{\mu}}) \to_{\texttt{surj}} E^{\tilde{\mu}}$.

■