Math. Struct. in Comp. Science (2001), *vol.* 11, *pp.* 207–260. Printed in the United Kingdom
© 2001 Cambridge University Press

Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus

PETER SELINGER

Department of Mathematics,
University of Michigan,
Ann Arbor, MI 48109-1109, U.S.A.
Email: selinger@umich.edu

Received 12 September 1998; revised 2 November 1999

We give a categorical semantics to the call-by-name and call-by-value versions of Parigot's $\lambda\mu$-calculus with disjunction types. We introduce the class of control categories, which combine a cartesian-closed structure with a premonoidal structure in the sense of Power and Robinson. We prove, via a categorical structure theorem, that the categorical semantics is equivalent to a CPS semantics in the style of Hofmann and Streicher. We show that the call-by-name $\lambda\mu$-calculus forms an internal language for control categories, and that the call-by-value $\lambda\mu$-calculus forms an internal language for the dual co-control categories. As a corollary, we obtain a syntactic duality result: there exist syntactic translations between call-by-name and call-by-value which are mutually inverse and which preserve the operational semantics. This answers a question of Streicher and Reus.

1. Introduction

The discussion about the relative advantages and disadvantages of the two parameter passing techniques, call-by-name and call-by-value, is almost as old as the theory of programming languages itself. While many modern functional programming languages use the call-by-value paradigm, which is easy to implement and semantically intuitive, Felleisen and Hieb write in their "Revised report on the syntactic theories of sequential control and state" that there is "no *theoretical* reason for choosing one over the other, even in the presence of control operators and assignments" (Felleisen and Hieb 1992).

In this paper, we study the relationship between the call-by-name and call-by-value paradigms for Parigot's $\lambda\mu$-calculus. The $\lambda\mu$-calculus is an extension of the simply-typed lambda calculus with certain sequential control operators. We show that, in the presence of product and disjunction types, the call-by-name and call-by-value $\lambda\mu$-calculi are *isomorphic* to each other, in the sense that there exist syntactic translations between them that preserve the operational semantics and that are mutually inverse up to isomorphism of types. These translations take the form of a *duality*: they turn argument-driven computation into demand-driven computation by exchanging input and output throughout, turning terms "inside out". The presence of disjunction types makes this possible: we can regard a term $M : A_1 \wedge \ldots \wedge A_n \to B_1 \vee \ldots \vee B_m$ as a function in n arguments with m possible result types. Under the duality between call-by-value and call-by-name,

M is mapped to a function in m arguments with n possible result types. The existence of such a duality in the context of the $\lambda\mu$-calculus was conjectured by Streicher and Reus (1998).

An interesting aspect of this duality is that it exchanges functional and imperative features. For instance, a purely functional call-by-value term is mapped to a call-by-name term that relies almost exclusively on control operators, and vice versa. This observation suggests that, from a practical point of view, certain algorithms are more naturally formulated in a call-by-value paradigm, and others in call-by-name. It is interesting to compare this with Filinski's work, in which he obtains a duality result by working with a larger and more symmetric syntax, in which the dual of a term is essentially its mirror image (Filinski 1989).

The main contribution of this paper, and the basis for the above-mentioned duality result, is a sound and complete categorical semantics for both the call-by-name and call-by-value $\lambda\mu$-calculus. We introduce the class of *control categories*, in which the call-by-name $\lambda\mu$-calculus can be interpreted in much the same way as the simply-typed lambda calculus is interpreted in a cartesian-closed category. We prove a categorical structure theorem that shows that every control category is equivalent to a "category of continuations", in the sense of Hofmann and Streicher (1997). This structure theorem implies the soundness and completeness of the categorical interpretation of the $\lambda\mu$-calculus with respect to a natural CPS semantics. But more is true: we show that the call-by-name $\lambda\mu$-calculus forms an *internal language* for the class of control categories.

We then repeat this process for the call-by-value calculus. We show that the call-by-value $\lambda\mu$-calculus forms an internal language for the class of *co-control categories*, which are simply the categorical duals of control categories. The syntactic duality result is then a corollary of the syntax-free categorical duality.

It should be stressed that the results of this paper are not particular to the $\lambda\mu$-calculus. They apply equally well to other, more traditional languages with continuation-like control constructs, such as call/cc in ML or Scheme, or Felleisen's $\mathcal{C}$ operator (Felleisen 1986). Operationally, all these calculi are equivalent; for instance, the equivalence between the $\lambda\mu$-calculus and a call-by-name version of Felleisen's $\mathcal{C}$ was shown by De Groote (1994b). One of the reasons that we have chosen the $\lambda\mu$-calculus as the basis for the semantics in this paper is because it is technically convenient to work with two separate name spaces, and thus with two-sided sequents, rather than with explicit negation types. This two-sidedness also facilitates our statement of duality.

Related Work

This work draws on several recent developments in the categorical semantics of control operators. The starting point of our work is Hofmann and Streicher's categorical semantics of the call-by-name $\lambda\mu$-calculus in terms of categorical continuation models (Hofmann and Streicher 1997). Our control categories are an abstraction of these models. Unlike categories of continuations, control categories are defined as categories with algebraic structure, and they allow a covariant interpretation of the $\lambda\mu$-calculus without any explicit reference to continuations. The crucial ingredient in defining the structure of a control category is the realization that disjunction is not bifunctorial, but that it forms a premonoidal structure in the sense of Power and Robinson (1997). Our structure theorem relates this abstract approach to Hofmann and Streicher's more concrete semantics by showing that any control category is equivalent to a category of continuations.

In the call-by-value case, our model is almost identical to Thielecke's interpretation in a $\otimes\neg$-

category (Thielecke 1997). Indeed, a co-control category is a $\otimes\neg$-category with additional structure. Thielecke's semantics does not include disjunction types, maybe because they are not central to the computational phenomena and real-life programming languages that he is interested in modeling. However, for this present work, the disjunction types are crucial, because they are indispensable for the statement of duality. In particular, our call-by-name semantics is strictly different from that of Thielecke, and cannot be expressed purely in terms of $\otimes\neg$-categories. Also, the presence of disjunction types reveals the nature of Thielecke's "self-adjointness" property, which becomes a special instance of a co-cartesian-closed structure. A structure theorem similar to ours was shown independently by Führmann (1998) for the case of $\otimes\neg$-categories. Also, in a more recent development, Führmann has given a more general account of the relationship between direct and monadic models, which generalizes some aspects of the present work to arbitrary computational effects in place of continuations (Führmann 1999).

A different class of models for the call-by-name $\lambda\mu$-calculus, based on fibrations, was defined by Ong and Ritter and later generalized to the disjunctive case by Pym and Ritter (Ong 1996; Pym and Ritter 1998). The focus of these models is different from ours, as they stress the fibered nature of the $\lambda\mu$-calculus with respect to control contexts, and thus they are, in a sense, higher-order. However, these models are rich in algebraic structure, and indeed, the $\lambda\mu$-calculus forms an internal language for them, in the suitable fibered sense. One may go back and forth between Ong/Ritter models and control categories by identifying the object A at the fiber Δ with the object $A \parr \Delta$ in a control category. This appears to be an instance of a more general construction of obtaining a fibration from a premonoidal structure, see also (Power and Robinson 1997).

Sometimes the question is raised what, if anything, is the computational significance of the disjunction types in the $\lambda\mu$-calculus. The question arises because these types are originally motivated mainly by logical and categorical concerns, and not by computational considerations. But it turns out that the disjunction types do indeed have a computational interpretation, in terms of certain manipulations with stacks. This is best seen in an abstract machine model. From the CPS semantics of this paper, one can derive a Krivine-style abstract machine, as was done for the fragment without disjunction in (Streicher and Reus 1998). The abstract machine model for the disjunctive call-by-name $\lambda\mu$-calculus, and an implementation, is described in detail in a separate paper (Selinger 1998). For the purposes of this present paper, we emphasize the logical and categorical perspective.

Outline

In Sections 2 through 4, we introduce control categories and exhibit their basic structure. In Sections 5 through 7, we discuss the interpretation of the call-by-name and call-by-value $\lambda\mu$-calculi. In Section 8, we discuss duality. Some technical proofs from Section 3 are given in the Appendix.

2. Control categories

In a cartesian-closed category, we use the notation $\diamond_A : A \to 1$ for the terminal arrow, π_1, π_2 for the first and second projections, $\langle f, g\rangle$ for pairing, $\epsilon_{A,B} : B^A \times A \to B$ for application, and $f^\star : B \to C^A$ for the curry of a map $f : B \times A \to C$. We also use the internal language of a

ccc to denote morphisms: thus a morphism $f : A \to B$ will be denoted by a typing judgment $x{:}A \rhd M : B$ in the usual way. Obvious subscripts are often omitted. Sometimes, we use the notation $\xrightarrow{ccc}$ to label evident ccc isomorphisms.

2.1. *Premonoidal categories*

Premonoidal categories were introduced by Power and Robinson (1997). We summarize the definition here. A premonoidal category is similar to a monoidal category, except that the tensor product is only assumed to be functorial in each argument *separately*, but not necessarily *jointly*. Thus, the tensor product in a premonoidal category is not in general bifunctorial; for lack of a better term we call such an operation a binoidal functor. The formal definition follows, where $|\mathbf{A}|$ denotes the class of objects of a category $\mathbf{A}$, regarded as a discrete subcategory.

Definition 2.1. Let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ be categories. A ***binoidal functor*** $F : \mathbf{A} \otimes \mathbf{B} \to \mathbf{C}$ is given by two bifunctors $F_0 : \mathbf{A} \times |\mathbf{B}| \to \mathbf{C}$ and $F_1 : |\mathbf{A}| \times \mathbf{B} \to \mathbf{C}$, such that $F_0(A, B) = F_1(A, B)$ for all pairs of objects A, B.

Since F_0 and F_1 agree where they are both defined (namely on objects), there is no harm in denoting both of them by F and thus writing $F(A, B)$, $F(f, B)$, and $F(A, g)$, where A, B are objects and f, g are morphisms. However, it does not in general make sense to write $F(f, g)$, because the two composites $F(f, B') \circ F(A, g)$ and $F(A', g) \circ F(f, B)$ may not coincide. A bifunctor is just a binoidal functor where the latter two compositions are equal.

The notation $F : \mathbf{A} \otimes \mathbf{B} \to \mathbf{C}$ is justified because the following pushout defines a tensor product in **Cat**:

$$\begin{array}{ccc} |\mathbf{A}| \times |\mathbf{B}| & \longrightarrow & \mathbf{A} \times |\mathbf{B}| \\ \downarrow & & \downarrow \\ |\mathbf{A}| \times \mathbf{B} & \longrightarrow & \mathbf{A} \otimes \mathbf{B} \end{array}$$

Thus, a binoidal functor can be regarded as a functor from $\mathbf{A} \otimes \mathbf{B}$ to $\mathbf{C}$. An explicit description of the category $\mathbf{A} \otimes \mathbf{B}$ is given in (Power and Robinson 1997). More generally, we can define n-oidal functors $F : \mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_n \to \mathbf{C}$ for every n.

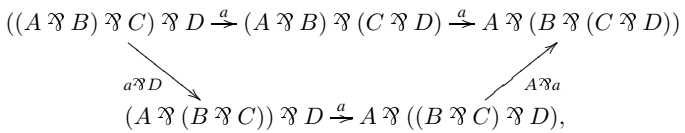
When we speak of natural transformations between binoidal functors, we always mean transformations that are natural in each component separately. For bifunctors, this coincides with the usual definition.

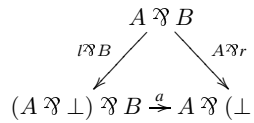
Definition 2.2. A ***binoidal category*** is a category $\mathbf{P}$ together with a binoidal functor $\parr : \mathbf{P} \otimes \mathbf{P} \to \mathbf{P}$. We use the usual infix notation $A \parr B$. A morphism $f : A \to A'$ in a binoidal category is ***central*** if for every morphism $g : B \to B'$, the two composites $(f \parr B') \circ (A \parr g)$ and $(A' \parr g) \circ (f \parr B)$ agree, and the two composites $(B' \parr f) \circ (g \parr A)$ and $(g \parr A') \circ (B \parr f)$ agree. In this case, we also use the notation $f \parr g$, respectively, $g \parr f$.

Premonoidal categories are defined by analogy with monoidal categories. Notice that the structural isomorphisms are required to be central.

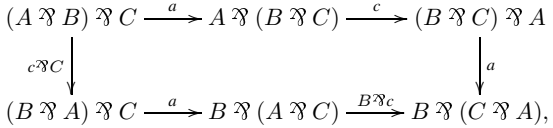
Definition 2.3. A ***premonoidal category*** is a binoidal category $\mathbf{P}$, together with an object $\bot$ and central natural isomorphisms $a_{A,B,C} : (A \parr B) \parr C \to A \parr (B \parr C)$, $l_A : A \to A \parr \bot$, and

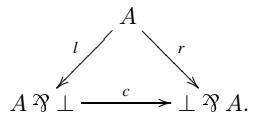
$r_A : A \to \bot \parr A$, subject to the usual coherence conditions:





A ***symmetric premonoidal category*** has in addition a family of central natural isomorphisms $c_{A,B} : A \parr B \to B \parr A$, satisfying $c \circ c = id$ and coherence:





The operation $\parr$ is also called a (symmetric) ***pretensor***.

The central morphisms of a premonoidal category $\mathbf{P}$ form a monoidal subcategory, called the ***center*** of $\mathbf{P}$, and denoted by $\mathbf{P}^\bullet$. Clearly, the center is the largest subcategory on which $\parr$ restricts to a proper (bifunctorial) tensor product. Coherence for premonoidal categories follows easily from Mac Lane's result for monoidal categories (Mac Lane 1963; Kelly 1964), since all the relevant coherence diagrams are contained in the center.

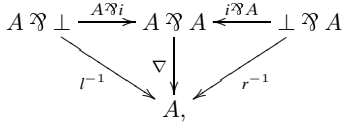
Premonoidal categories share many properties of monoidal categories, but some special care is necessary when manipulating them. For instance, one should keep in mind that there are innocent-looking expressions, such as $A \parr A$, that are not functorial. Also notice that if $f : A \to A'$ is a morphism, then the induced family of arrows $A \parr B \to A' \parr B$ is not in general natural in B. We say that a family of maps $\eta_B : F(B) \to G(B)$ is ***natural in central B*** if it is natural with respect to central $g : B \to B'$, and analogously for dinaturality.

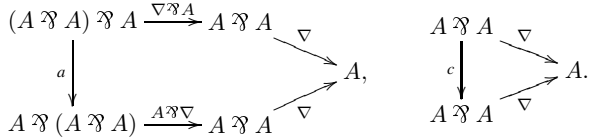
A remark on the choice of symbols: I originally chose the upside-down ampersand "$\parr$" because it suggests a tensor product with a disjunctive flavor. I did not intend to imply a connection to linear logic by this choice. However, in recent work with O. Laurent and L. Regnier, it turned out that control categories are a model of proof-nets for polarized linear logic, and, to my surprise, the connective "$\parr$" indeed corresponds to the "par" of linear logic under this interpretation. A more detailed account of this connection will appear elsewhere.

2.2. *Codiagonals and focus*

Definition 2.4. Let $\mathbf{P}$ be a symmetric premonoidal category. A ***symmetric monoid*** in $\mathbf{P}$ is given by an object A, together with two central morphisms $i_A : \bot \to A$ and $\nabla_A : A \parr A \to A$,

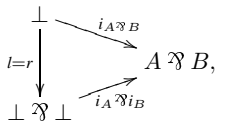
satisfying the usual equations:

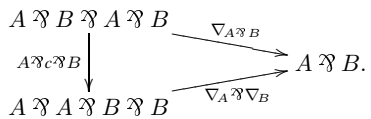




A symmetric premonoidal category ***has codiagonals*** if there is a chosen symmetric monoid $\langle A, i_A, \nabla_A \rangle$ for each object A, compatible with the premonoidal structure in the following sense:

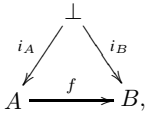
$i_\bot = id_\bot : \bot \to \bot$,



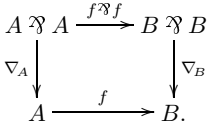


In the last diagram, some obvious associativity isomorphisms have been omitted. Since every premonoidal category can be shown to be equivalent to a strict one (Power and Robinson 1997), we will henceforth and without loss of generality treat associativity as if it were an identity map.

Notice we do not require that the families of maps $i_A : \bot \to A$ and $\nabla_A : A \parr A \to A$ are natural in A; in fact, it is not even obvious how one would state the naturality of ∇_A. Instead, we will call a central morphism $f : A \to B$ ***discardable*** if



and ***copyable*** if



This terminology is taken from (Thielecke 1997). Strictly speaking, we should use the terms *co-discardable* and *co-copyable*, but this would be cumbersome. Discardability and copyability, like centrality, are notions of "well-behavedness".

Definition 2.5. A morphism is called ***focal*** if it is central, discardable, and copyable.

Remark 2.6. The focal morphisms form a subcategory of $\mathbf{P}$, called the ***focus*** of $\mathbf{P}$, which we denote by $\mathbf{P}^\sharp$. The focus is contained in the center. It is closed under $\parr$. All the structural maps (a, l, r, c, i, and ∇) are focal, and so are the left and right weakening maps defined by

$$\begin{aligned} w^l_{A,B} &= A \xrightarrow{l} A \parr \bot \xrightarrow{A \parr i_B} A \parr B, \\ w^r_{A,B} &= B \xrightarrow{r} \bot \parr B \xrightarrow{i_A \parr B} A \parr B. \end{aligned}$$

Sometimes, we denote either of these maps by w. The focus has a canonical finite coproduct structure:

Lemma 2.7. *In $\mathbf{P}^\sharp$, the object $\bot$ is initial, and $\parr$ is a coproduct with the following injections and co-pairing:*

$$\begin{aligned} \mathit{inl} &= A \xrightarrow{w^l} A \parr B, \\ \mathit{inr} &= B \xrightarrow{w^r} A \parr B, \\ [f,g] &= A \parr B \xrightarrow{f \parr g} C \parr C \xrightarrow{\nabla_C} C. \end{aligned}$$

In fact, $\mathbf{P}^\sharp$ is the largest subcategory of $\mathbf{P}$ on which $\parr$ restricts to a coproduct.

Remark 2.8. One has $\mathbf{P}^\sharp \subseteq \mathbf{P}^\bullet \subseteq \mathbf{P}$. In general, the focus of a symmetric premonoidal category with codiagonals is strictly contained in the center: for instance, if $\mathbf{P}$ is a monoidal category where the tensor is not given by a coproduct, then the the center is all of $\mathbf{P}$, whereas the focus is not. However, in the case of a control category, to be defined shortly, we will see that the center and the focus always coincide.

2.3. *Distributivity*

Definition 2.9. Suppose $\mathbf{P}$ is a symmetric premonoidal category with codiagonals. Suppose that $\mathbf{P}$ also has finite products. We say that $\mathbf{P}$ is ***distributive*** if the projections $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ are focal, and if for all objects C, the functor $- \parr C$ preserves finite products.

Note that the functor $- \parr C$ preserves finite products iff for all objects A, B, and C, the natural maps

$$(A \times B) \parr C \xrightarrow{\langle \pi_1 \parr C, \pi_2 \parr C \rangle} (A \parr C) \times (B \parr C) \qquad \text{and} \qquad 1 \parr C \xrightarrow{\diamond} 1$$

are isomorphisms. We denote the inverse of the first map by $d_{A,B,C} : (A \parr C) \times (B \parr C) \to (A \times B) \parr C$.

Lemma 2.10. *If $\mathbf{P}$ is a distributive, symmetric premonoidal category with codiagonals, then the focus of $\mathbf{P}$ is closed under the finite product structure.*

Proof. First, it is trivial to see that $\diamond_A : A \to 1$ is focal. Second, whenever $f : C \to A$ and $g : C \to B$ are central and $h : D \to E$, then

$$\begin{array}{ccccc} C \parr D & \xrightarrow{\langle f \parr D, g \parr D \rangle} & (A \parr D) \times (B \parr D) & \xrightarrow{d} & (A \times B) \parr D \\ \downarrow{\scriptstyle C \parr h} & & \downarrow{\scriptstyle (A \parr h) \times (B \parr h)} & & \downarrow{\scriptstyle (A \times B) \parr h} \\ C \parr E & \xrightarrow{\langle f \parr E, g \parr E \rangle} & (A \parr E) \times (B \parr E) & \xrightarrow{d} & (A \times B) \parr E. \end{array}$$

The left square commutes by hypothesis, the right one by naturality of d. It follows from the definition of d that the map along the top is $\langle f, g \rangle \parr D$, and similarly along the bottom. Thus, the perimeter of the diagram shows that $\langle f, g \rangle$ is central. Next, assume that f and g are also discardable and copyable. The commutativity of the following two diagrams follows by post-composing each of them with π_1 and with π_2, and by using the fact that π_1 and π_2 are focal.

Thus, $\langle f, g \rangle$ is discardable and copyable as well.

$$\begin{array}{ccc} & \bot & \\ {\scriptstyle i_C} \swarrow & & \searrow {\scriptstyle i_{A \times B}} \\ C & \xrightarrow{\langle f,g \rangle} & A \times B \end{array} \qquad \begin{array}{ccc} C \parr C & \xrightarrow{\langle f,g \rangle \parr \langle f,g \rangle} & (A \times B) \parr (A \times B) \\ \downarrow{\scriptstyle \nabla_C} & & \downarrow{\scriptstyle \nabla_{A \times B}} \\ C & \xrightarrow{\langle f,g \rangle} & A \times B. \end{array}$$

□

Notice that since d^{-1} was defined in terms of pairing, the lemma also implies that d^{-1}, and thus d, is focal.

2.4. *Control categories*

To the structure that we have considered so far, we now add cartesian-closedness, along with some conditions that relate the cartesian-closed structure and the premonoidal structure.

Definition 2.11. Suppose $\mathbf{P}$ is a distributive symmetric premonoidal category with codiagonals. Suppose $\mathbf{P}$ is also cartesian-closed. Let $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ be the canonical morphism obtained by currying

$$\hat{\epsilon}_{A,B,C} : (B^A \parr C) \times A \xrightarrow{(B^A \parr C) \times w} (B^A \parr C) \times (A \parr C) \xrightarrow{d} (B^A \times A) \parr C \xrightarrow{\epsilon \parr C} B \parr C.$$

The category $\mathbf{P}$ is called a ***control category*** if $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ is a natural isomorphism in A, B, and C, satisfying the following coherence conditions:

1. The following diagram commutes, where $s'_{A,B,C} = B \parr C^A \xrightarrow{c} C^A \parr B \xrightarrow{s_{A,C,B}} (C \parr B)^A \xrightarrow{c^A} (B \parr C)^A$,

$$\begin{array}{ccc} B^A \parr C^D & \xrightarrow{s'} & (B^A \parr C)^D \\ \downarrow{\scriptstyle s} & & \downarrow{\scriptstyle s^D} \\ (B \parr C^D)^A & \xrightarrow{s'^A} & ((B \parr C)^D)^A \xrightarrow{ccc} ((B \parr C)^A)^D. \end{array}$$

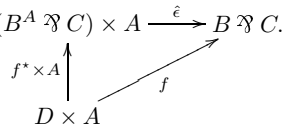
2. The following two diagrams commute, where $\Delta_A : A \to A \times A$ is $\langle id_A, id_A \rangle$:

$$\begin{array}{ccccc} B^A \parr B^A & \xrightarrow{s'} & (B^A \parr B)^A & \xrightarrow{s^A} & (B \parr B)^{A \times A} \\ & {\scriptstyle \nabla_{B^A}} \searrow & & \swarrow {\scriptstyle \nabla_B{}^{\Delta_A}} & \\ & & B^A, & & \end{array} \qquad \begin{array}{ccc} \bot & \xrightarrow{ccc} & \bot^1 \\ {\scriptstyle i_{B^A}} \searrow & & \swarrow {\scriptstyle (i_B)^{\diamond_A}} \\ & B^A. & \end{array}$$

Remark 2.12. While it automatically follows from the definition of $s_{A,B,C}$ that it is natural in A and B, the requirement that it is natural in C is needed as a separate axiom.

The isomorphism $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ is called ***exponential strength***. The requirement that s is an isomorphism is equivalent to the requirement that for every $f : D \times A \to B \parr C$,

there exists a unique $f^\star : D \to B^A \parr C$ such that



Thus, one has a natural isomorphism of hom-sets $(D \times A, B \parr C) \cong (D, B^A \parr C)$, giving rise to a *fibered ccc-structure* on $\mathbf{P}$.

Remark 2.13. So far, all the structural maps (of the premonoidal structure, the codiagonals, and the finite product structure) have been focal. However, we do not require the application map $\epsilon : B^A \times A \to B$ to be focal, nor for the focus to be closed under currying. On the other hand, the exponential strength $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ turns out to be focal, as we will show in Lemma 3.5 below.

2.5. *Example: Categories of continuations*

As an example of a control category, we consider a category of continuations in the style of Hofmann and Streicher (1997). We begin with a category $\mathbf{C}$ with distributive finite products and coproducts, and with a distinguished object R, such that for all objects A, an exponential R^A exists. For example, one may take $\mathbf{C}$ to be a bicartesian closed category (Lambek and Scott 1986), although in general, we do not require arbitrary exponentials to exist. We say that $\mathbf{C}$ satisfies the ***mono requirement*** if the canonical morphism $\partial_A : A \to R^{R^A}$ is monic for all A. In this case, we call the category $\mathbf{C}$ a ***response category***, and the object R its ***object of responses***. This terminology is borrowed from continuation semantics. For simplicity and without loss of generality, we assume that the exponentials are chosen such that $A \neq B$ implies $R^A \neq R^B$.

Given a response category $\mathbf{C}$, we define its ***category of continuations***, denoted $R^{\mathbf{C}}$, to be the full subcategory of $\mathbf{C}$ consisting of the objects of the form R^A. The crucial observation underlying continuation semantics is that the category $R^{\mathbf{C}}$ is cartesian-closed (Agapiev and Moggi 1991; Lafont, Reus, and Streicher 1993). Indeed, in $\mathbf{C}$, one has

$$1 \cong R^0, \qquad R^A \times R^B \cong R^{A+B}, \qquad (R^B)^{R^A} \cong R^{B \times R^A},$$

and, being a full subcategory, $R^{\mathbf{C}}$ inherits this structure from $\mathbf{C}$. Moreover, the category $R^{\mathbf{C}}$ has a canonical premonoidal structure, given on objects by

$$\bot := R^1 \cong R, \qquad R^A \parr R^B := R^{A \times B}.$$

The operation $\parr$ is functorial in the first argument via $R^{A \times B} \cong (R^A)^B$, and in the second argument via $R^{A \times B} \cong (R^B)^A$. Notice that the operation $\parr$ is not functorial in both arguments jointly. All maps of the form R^f are focal. The structural maps *a*, *l*, *r*, *c*, *i*, and ∇ are defined in the obvious way.

Lemma 2.14. *The category of continuations $R^{\mathbf{C}}$ is a control category.*

Proof. The axioms are easily checked. For instance, exponential strength holds because

$$(R^B)^{R^A} \parr R^C \cong R^{B \times R^A \times C} \cong (R^B \parr R^C)^{R^A}.$$

□

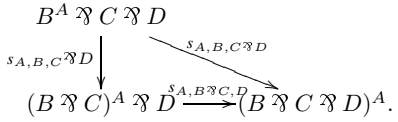
A converse is also true: in Section 3.9, we will prove the main theorem about control categories: every control category is equivalent to a category of continuations.

3. The structure of a control category

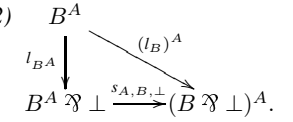
3.1. *Coherence*

Lemma 3.1. *The following hold in a control category:*

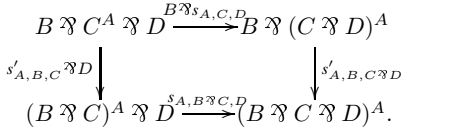
(1)



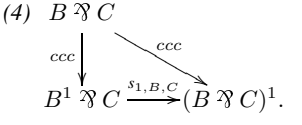
(2)



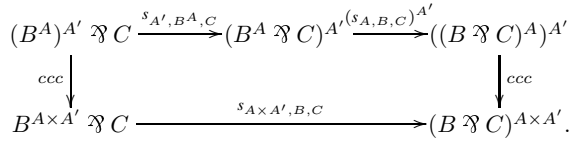
(3)



(4)



(5)



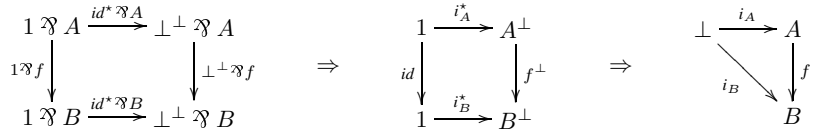
Proof. See appendix.

3.2. *Centrality and discardability*

We show here that any central map is discardable. In Section 3.6, we will be able to show that any central map is also copyable, and thus the center and the focus coincide in a control category.

Lemma 3.2. *In a control category, any central morphism is discardable.*

Proof. Let $f : A \to B$ be central. Let $id^\star : 1 \to \bot^\bot$ be the curry of the identity map. Then the first of the following three diagrams commutes by centrality. The second diagram is the same as the first one up to coherent isomorphisms, and thus it also commutes. Finally, the third diagram is obtained by uncurrying. Thus, f is discardable.

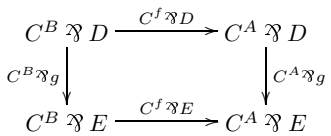


□

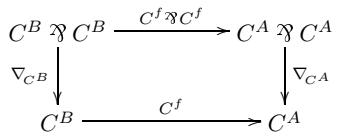
3.3. *Some focal structural maps*

Lemma 3.3. *In a control category, any morphism of the form $C^f : C^B \to C^A$ is focal, where $f : A \to B$.*

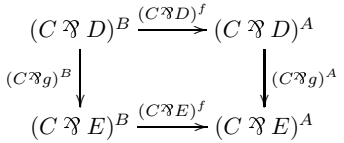
Proof. The commutativity of the two diagrams



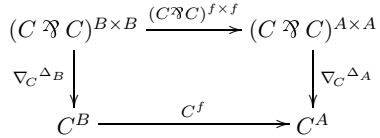
and



follows from that of



and



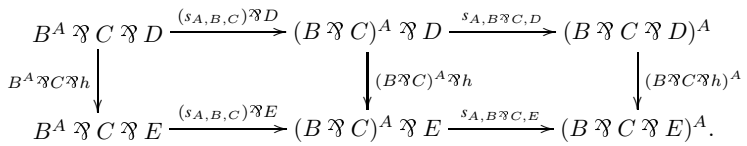
respectively, by naturality of $C^B \parr D \cong (C \parr D)^B$ and of $C^B \parr C^B \cong (C \parr C)^{B\times B}$. □

Lemma 3.4. *In a control category, the natural ccc isomorphisms $B^1 \cong B$ and $(B^A)^{A'} \cong B^{A\times A'}$ are focal.*

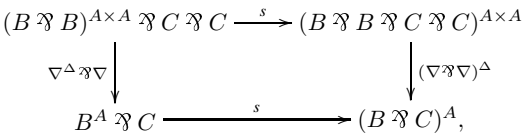
Proof. This follows from Lemma 3.1(4) and (5). □

Lemma 3.5. *In a control category, the exponential strength $s_{A,B,C} : B^A \parr C \to (B \parr C)^A$ is focal.*

Proof. To see that it is central, consider the diagram



By coherence, the morphism along the top is equal to $s_{A,B,C\parr D}$, and similarly along the bottom. The right square commutes by naturality of strength, and so does the perimeter. The left square, then, implies that $s_{A,B,C}$ is central. Showing that $s_{A,B,C}$ is copyable comes down, modulo coherence, to showing that

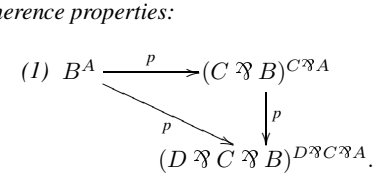


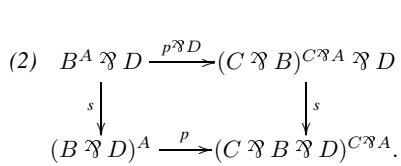
which follows by naturality of s. □

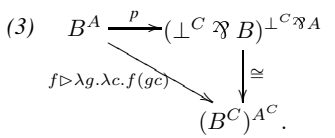
Lemma 3.6. *For all A, B, C, the canonical morphism $p_{A,B,C} : B^A \to (C \parr B)^{C\parr A}$, which is obtained by currying*

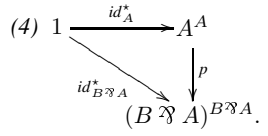
$$\tilde{\epsilon}_{A,B,C} : B^A \times (C \parr A) \xrightarrow{w\times(C\parr A)} (C \parr B^A) \times (C \parr A) \xrightarrow{d} C \parr (B^A \times A) \xrightarrow{C\parr\epsilon} C \parr B,$$

is focal. Moreover, p is natural in A, B and dinatural in central C, and it satisfies the following coherence properties:

(1) 

(2) 

(3) 

(4) 

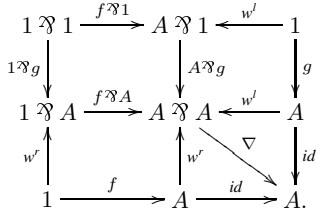
Proof. See appendix.

3.4. *A remark on consistency*

One may ask whether it is consistent to trivialize the structure of $\parr$, i.e. to assume that $\parr$ is bifunctorial. It turns out that any control category in which $\parr$ is bifunctorial is equivalent to a boolean algebra. The reader may find it instructive to compare the following lemma to the fact, proved in (Lambek and Scott 1986, p.67), that in a bicartesian closed category, there is no arrow $A \to 0$ unless $A \cong 0$.

Lemma 3.7. *There is no central morphism $f : 1 \to A$, unless $A \cong 1$.*

Proof. First, we claim that if $f, g : 1 \to A$ and f is central, then $f = g$. Consider the diagram



All cells commute, and the morphisms along the top and left sides are isomorphisms. Hence it follows that $f = g$, proving the first claim. Now suppose that $f : 1 \to A$ is central, and let B be any object. Then $f^\diamond : 1 \cong 1^1 \to A^B$ is central by Lemma 3.3. By our first claim, the hom-set $(1, A^B) \cong 1$, and hence $(B, A) \cong 1$, showing that A is terminal. □

Corollary 3.8. *A control category in which $\parr$ is bifunctorial is equivalent to a boolean algebra.*

Proof. If $\parr$ is bifunctorial, then all morphisms are central. Thus, any hom-set $(B, A) \cong (1, A^B)$ has at most one element by Lemma 3.7. It follows that the category is equivalent to a poset, and the control category structure trivializes to a boolean algebra structure. □

3.5. *Classical features: excluded middle and double negation*

It is well-known that lambda calculi with control operators, such as the $\lambda\mu$-calculus, correspond to classical logic under a propositions-as-types correspondence. This fact was first discovered by Griffin (1990). Since control categories are going to be models for such calculi, it should therefore not be surprising that control categories are models of propositional classical logic. Objects correspond to propositions, and arrows correspond to proofs. The operation $\parr$ models disjunction. Note that all the axioms of control categories are intuitionistically valid, except for the existence of an inverse to the map $s : B^A \parr C \to (B \parr C)^A$. The latter makes the logic classical. We can define arrows for excluded middle and double negation:

$$
\begin{array}{ll}
tnd_A : 1 \xrightarrow{r^\star_A} (\bot \parr A)^A \xrightarrow{s^{-1}} \bot^A \parr A & \textit{(excluded middle)} \\
\partial_A : A \xrightarrow{\epsilon^\star} \bot^{\bot^A} & \textit{(double negation introduction)} \\
\theta_A : \bot^{\bot^A} \xrightarrow{p} (A \parr \bot)^{A \parr \bot^A} \xrightarrow{\cong} A^{A \parr \bot^A} \xrightarrow{A^{tnd}} A & \textit{(double negation elimination)}
\end{array}
$$

Of course, ∂_A is just the natural map that can be defined in any ccc.

Lemma 3.9. *In a non-trivial control category:*

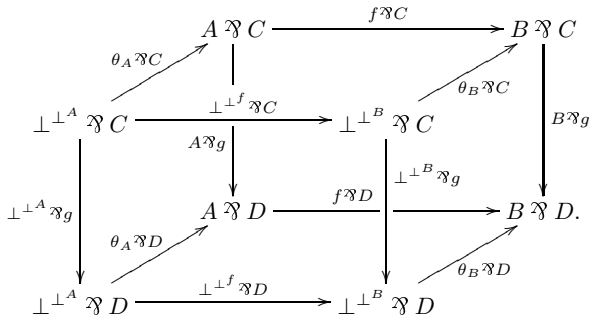
(1) $\theta_{\bot^A} = \bot^{\partial_A} : \bot^{\bot^{\bot^A}} \to \bot^A$.
(2) $\theta_A \circ \partial_A = id_A : A \to A$.
(3) tnd_A *is dinatural in* A*, but not in general central.*
(4) ∂_A *is natural in* A*, but not in general central.*
(5) θ_A *is focal, but not natural in* A*. However,* θ_A *is natural in central* A*.*

Proof. (1): This follows from Lemma 3.6(3). (2): An easy diagram chase. (3): Dinaturality follows from naturality of *r* and *s*. Notice that by Lemma 3.7, tnd_A is not central unless $\bot^A \parr A \cong 1$. (4): The map ∂_A is natural in any ccc. If $\partial_A : A \to \bot^{\bot^A}$ is central, then any map $f : A \to B$ is central, because $f = \theta_B \circ \partial_B \circ f = \theta_B \circ \bot^{\bot^f} \circ \partial_A$ by (2), which is central by (5) and Lemma 3.3. (5): θ_A is focal by its definition, because *p* is focal by Lemma 3.6 and A^{tnd} is focal by Lemma 3.3. The naturality of θ_A in central A follows from the dinaturality of *tnd* and *p* in central A. □

The central morphisms are characterized by the fact that θ_A is natural in central A:

Lemma 3.10. *A morphism* $f : A \to B$ *is central if and only if* $f \circ \theta_A = \theta_B \circ \bot^{\bot^f}$.

Proof. "Only if" follows from Lemma 3.9. For "if", suppose $f \circ \theta_A = \theta_B \circ \bot^{\bot^f}$, and let $g : C \to D$. Consider the following cube:



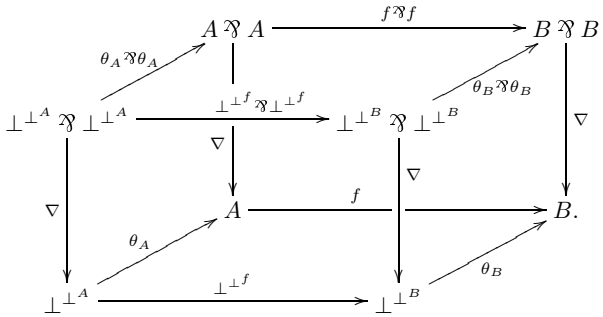
The top and bottom faces commute by assumption. The left, right, and front faces commute because θ_A, θ_B, and $\bot^{\bot^f}$ are central. Moreover, the top left arrow $\theta_A \parr C$ is a split epic by Lemma 3.9(2), and thus the back face commutes, showing that f is central. □

3.6. *The center and the focus coincide*

From Lemma 3.2, we know that any central map is discardable. We can now show that in a control category, the center and the focus coincide:

Lemma 3.11. *In a control category, any central morphism is copyable.*

Proof. The proof is adopted from Thielecke (1997). Suppose $f : A \to B$ is central. Consider the following cube:



The front face and the two sides commute because $\bot^{\bot^f}$, θ_A, and θ_B are copyable by Lemmas 3.3 and 3.9. The top and bottom faces commute because θ_A is natural in central A by Lemma 3.9. Moreover, the top left arrow $\theta_A \parr \theta_A$ is a split epic with right inverse $(\partial_A \parr \bot^{\bot^A}) \circ (A \parr \partial_A)$. Thus, it follows that the back face commutes, showing that f is copyable. □

3.7. *The basic adjunctions of the center*

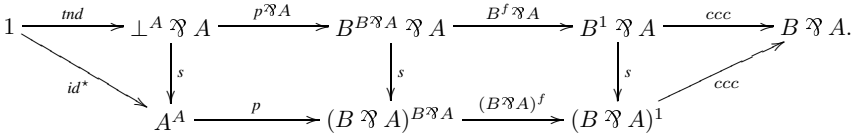
Let $\mathbf{P}$ be a control category. Recall that the center of $\mathbf{P}$ was denoted by $\mathbf{P}^\bullet$. We use the usual notation for hom-sets. Thus, $\mathbf{P}(A, B)$ is the set of all morphisms, and $\mathbf{P}^\bullet(A, B)$ is the set of central morphisms from A to B.

Lemma 3.12. $\mathbf{P}(1, B \parr A) \cong \mathbf{P}^\bullet(\bot^A, B)$*, naturally in* A *and central* B*.*

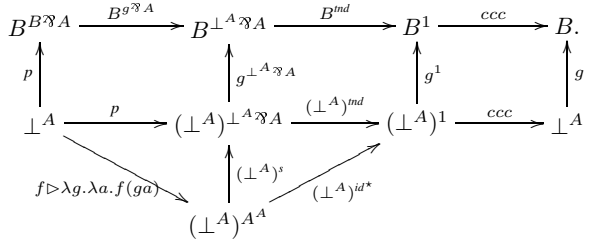
Proof. Writing $p_{A,B}$ for $\bot^A \xrightarrow{p_{A,\bot,B}} (B \parr \bot)^{B \parr A} \xrightarrow{\cong} B^{B \parr A}$, we define $\phi_{A,B} : \mathbf{P}(1, B \parr A) \to \mathbf{P}^\bullet(\bot^A, B)$ and $\psi_{A,B} : \mathbf{P}^\bullet(\bot^A, B) \to \mathbf{P}(1, B \parr A)$ by

$$\begin{array}{rcl} \phi_{A,B}(f) & = & \bot^A \xrightarrow{p_{A,B}} B^{B \parr A} \xrightarrow{B^f} B^1 \xrightarrow{ccc} B, \\ \psi_{A,B}(g) & = & 1 \xrightarrow{tnd_A} \bot^A \parr A \xrightarrow{g \parr A} B \parr A. \end{array}$$

Notice that $\phi(f)$ is indeed central by Lemmas 3.3, 3.4, and 3.6. ψ is clearly natural in central B. The naturality of ϕ in A follows from that of $p_{A,B}$. To see that $\psi(\phi(f)) = f$ holds for all $f : 1 \to B \parr A$, consider the following diagram:



The diagram commutes, from left to right, by definition of *tnd*, Lemma 3.6(2), naturality of *s*, and Lemma 3.1(4), respectively. The composition along the top is $\psi(\phi(f))$. The composition along the bottom is f by Lemma 3.6(4) and standard ccc manipulations. To show that $\phi(\psi(g)) = g$ holds for central g, consider



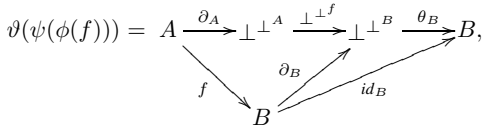
The leftmost square commutes by dinaturality of p in central g. The leftmost triangle commutes by Lemma 3.6. The other parts commute by definition of *tnd* and by ccc calculations. Clockwise along the top, we have $\phi(\psi(g))$, and counterclockwise along the bottom, we have g. □

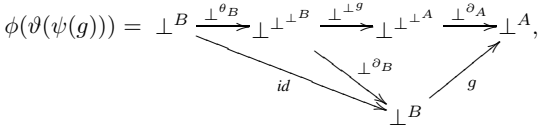
Lemma 3.13. $\mathbf{P}(A, B) \cong \mathbf{P}^\bullet(\bot^B, \bot^A) \cong \mathbf{P}^\bullet(\bot^{\bot^A}, B)$*. The first isomorphism is natural in* A *and* B*, and the second isomorphism is natural in* A *in central* B*.*

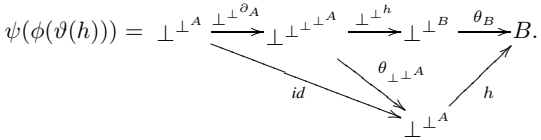
Proof. Define

$$\begin{array}{lllll} \phi_{A,B} : \mathbf{P}(A, B) & \to \mathbf{P}^\bullet(\bot^B, \bot^A) & \text{by} & \phi(f) & = & \bot^B \xrightarrow{\bot^f} \bot^A, \\ \psi_{A,B} : \mathbf{P}^\bullet(\bot^B, \bot^A) & \to \mathbf{P}^\bullet(\bot^{\bot^A}, B) & \text{by} & \psi(g) & = & \bot^{\bot^A} \xrightarrow{\bot^g} \bot^{\bot^B} \xrightarrow{\theta_B} B, \\ \vartheta_{A,B} : \mathbf{P}^\bullet(\bot^{\bot^A}, B) & \to \mathbf{P}(A, B) & \text{by} & \vartheta(h) & = & A \xrightarrow{\partial_A} \bot^{\bot^A} \xrightarrow{h} B. \end{array}$$

Notice that $\phi(f)$ and $\psi(g)$ are indeed central, by Lemmas 3.3 and 3.9(5). Clearly, ϕ is natural in A and B. Moreover, ψ is natural in A and in central B because θ_B is. We need to show that all three maps are isomorphisms:







The commutativity of these diagrams follows from Lemma 3.9. □

Putting the last two lemmas together, we immediately get:

Corollary 3.14. $\mathbf{P}^\bullet(\bot^\bot, B \parr A) \cong \mathbf{P}(1, B \parr A) \cong \mathbf{P}^\bullet(\bot^A, B)$*, naturally in central* A *and central* B*.* □

3.8. *Functors and equivalences of control categories*

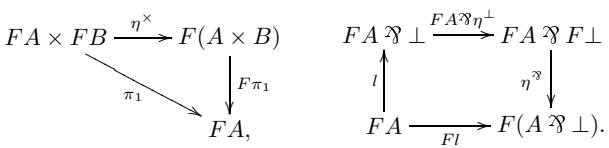
Let $\mathbf{P}$ and $\mathbf{P}'$ be control categories. A ***strict functor of control categories*** $F : \mathbf{P} \to \mathbf{P}'$ is a functor that preserves chosen structure, i.e., it preserves chosen binary products, $\parr$, 1, $\bot$, and exponentials, as well as the chosen morphisms associated with that structure. Notice that it follows from Lemma 3.10 that such a functor preserves central maps; thus, we do not need this as a special requirement.

In practice, we are usually more interested in functors that preserve the structure *up to isomorphism*. In the context of control categories, it is sensible to require the structure to be preserved up to *central* isomorphism, as expressed in the following definition:

Definition 3.15. A ***(weak) functor of control categories*** is a functor $F : \mathbf{P} \to \mathbf{P}'$, together with central natural isomorphisms

$$\begin{array}{lll} \eta^\times_{A,B} : & FA \times FB & \xrightarrow{\cong} F(A \times B) \\ \eta^\parr_{A,B} : & FA \parr FB & \xrightarrow{\cong} F(A \parr B) \\ \eta^1 : & 1 & \xrightarrow{\cong} F1 \\ \eta^\bot : & \bot & \xrightarrow{\cong} F\bot \\ \eta^{exp}_{A,B} : & FB^{FA} & \xrightarrow{\cong} F(B^A), \end{array}$$

commuting with the morphism structure in all the evident ways, for instance:



It follows from Lemma 3.10 that weak functors of control categories preserve the center. Note that this in particular implies that weak functors of control categories can be composed.

We will also need a notion of equivalence of control categories. Here, too, it is appropriate to modify the standard definition to take into account the concept of centrality.

Definition 3.16. An ***equivalence of control categories*** $\mathbf{P}$ and $\mathbf{P}'$ is given by a pair of weak functors of control categories, $F : \mathbf{P} \to \mathbf{P}'$ and $G : \mathbf{P}' \to \mathbf{P}$, together with two *central* natural isomorphisms $G \circ F \cong id_{\mathbf{P}}$ and $F \circ G \cong id_{\mathbf{P}'}$. If two control categories are equivalent in this sense, we also write $\mathbf{P} \simeq \mathbf{P}'$.

We say that a functor $F : \mathbf{P} \to \mathbf{P}'$ is ***centrally essentially onto objects*** if for each $B \in |\mathbf{P}'|$, there exists an $A \in |\mathbf{P}|$ and a central isomorphism $B \cong FA$.

Lemma 3.17. *Assuming the axiom of choice, a weak functor of control categories $F : \mathbf{P} \to \mathbf{P}'$ is part of an equivalence if and only if F is full, faithful, and centrally essentially onto objects.*

Proof. A very slight modification of the usual argument for categories with structure. □

3.9. *The structure theorem for control categories*

The fundamental theorem about control categories is the following structure theorem:

Theorem 3.18 (Structure Theorem). *Any control category $\mathbf{P}$ is equivalent to a category of continuations $R^{\mathbf{C}}$.*

Suppose $\mathbf{P}$ is a control category. Let $\mathbf{C} = (\mathbf{P}^\bullet)^{op}$ be the dual of the center of $\mathbf{P}$. Thus, the objects of $\mathbf{C}$ are those of $\mathbf{P}$, and a morphism in $\mathbf{C}$ from A to B is a central morphism from B to A in $\mathbf{P}$. To avoid confusion, we will write $\widehat{A}$ for the object A, when considered as an object of $\mathbf{C}$. Similarly, we will write $\widehat{f} : \widehat{A} \to \widehat{B}$ for a central morphism $f : B \to A$, when considered as a morphism of $\mathbf{C}$.

Lemma 3.19. *The category $\mathbf{C}$ has distributive finite products and coproducts.*

Proof. The center $\mathbf{P}^\bullet$ is closed under finite products by Lemma 2.10. Moreover, since center and focus coincide, the premonoidal structure of $\mathbf{P}$ restricts to a finite-coproduct structure on $\mathbf{P}^\bullet$ by Lemma 2.7. Thus, $\mathbf{C}$ has finite products and coproducts. The distributivity of $\mathbf{C}$ follows from that of $\mathbf{P}$. □

In $\mathbf{C}$, define an object of responses $R := \widehat{\bot^\bot}$.

Lemma 3.20. *The category $\mathbf{C}$ has exponentials of the form $R^{\widehat{A}}$ for every object $\widehat{A}$. Moreover, the canonical morphism $\partial_{\widehat{A}} : \widehat{A} \to R^{R^{\widehat{A}}}$ is monic. Thus, $\mathbf{C}$ is a response category.*

Proof. Using the natural equivalence of Corollary 3.14, we have

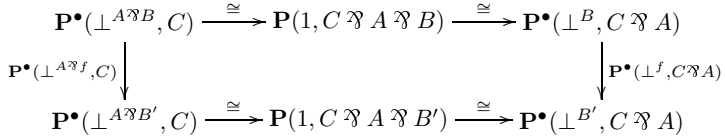
$$\mathbf{C}(\widehat{B} \times \widehat{A}, R) \cong \mathbf{P}^\bullet(\bot^\bot, B \parr A) \cong \mathbf{P}^\bullet(\bot^A, B) \cong \mathbf{C}(\widehat{B}, \widehat{\bot^A}),$$

naturally in $\widehat{A}$, $\widehat{B}$. Thus, we can define $R^{\widehat{A}} := \widehat{\bot^A}$ in $\mathbf{C}$. Moreover, $\partial_{\widehat{A}} : \widehat{A} \to R^{R^{\widehat{A}}}$ is $\widehat{\theta_A} : \widehat{A} \to \widehat{\bot^{\bot^A}}$, which is monic because θ_A is a split epic in $\mathbf{P}$ by Lemma 3.9(2). □

Proof of Theorem 3.18. It remains to be shown that the category of continuations $R^{\mathbf{C}}$ is equivalent to $\mathbf{P}$ as a control category. By Lemma 3.13, we know that the contravariant functor $F : \mathbf{P} \to \mathbf{P}^\bullet$ given by $F(A) = \bot^A$ and $F(f) = \bot^f$ is full and faithful. Thus, $F^{op} : \mathbf{P} \to \mathbf{C}$ is a full and faithful covariant functor. Moreover, the objects in the image of F^{op} are precisely those of the form $\widehat{\bot^A} = R^{\widehat{A}}$. Thus, F^{op} restricts to an equivalence of categories $\mathbf{P} \to R^{\mathbf{C}}$. We must show that it preserves control category structure. Being an equivalence, F^{op} clearly preserves finite products and exponentials. We calculate that it preserves the premonoidal structure:

$$F^{op}(A \parr B) = \widehat{\bot^{A \parr B}} = R^{\widehat{A} \times \widehat{B}} = R^{\widehat{A}} \parr R^{\widehat{B}} = \widehat{\bot^A} \parr \widehat{\bot^B} = F^{op}(A) \parr F^{op}(B),$$
$$F^{op}(\bot) = \widehat{\bot^\bot} = R = \bot.$$

One must also check that for $f : B \to B'$ in $\mathbf{P}$, one has $F^{op}(A \parr f) = F^{op}(A) \parr F^{op}(f)$, i.e., $\widehat{\bot^{A \parr f}} = \widehat{\bot^f}^{\widehat{A}} : R^{\widehat{A} \times \widehat{B}} \to R^{\widehat{A} \times \widehat{B'}}$. Unwinding the definition of exponentiation in $\mathbf{C}$, one finds that this holds if the following commutes:



But the isomorphisms, from Lemma 3.12, are natural in B, and thus this commutes. Finally, it is routine to check that F^{op} preserves the structural maps *a*, *l*, *r*, *c*, *i*, and ∇. This proves the theorem. □

4. More on the structure of a control category

In this section, we examine the structure of control categories further. The material of Subsections 4.1 and 4.2 is needed in preparation for the interpretation of the call-by-value $\lambda\mu$-calculus; the rest of this section may be skipped in the first reading.

4.1. *The weak co-closed structure of a control category*

By combining the cartesian-closed structure with Lemma 3.12, we get the following sequence of isomorphisms:

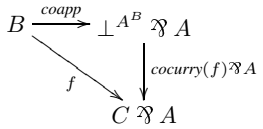
$$\mathbf{P}(B, C \parr A) \cong \mathbf{P}(1, (C \parr A)^B) \cong \mathbf{P}(1, C \parr A^B) \cong \mathbf{P}^\bullet(\bot^{A^B}, C),$$

naturally in A, B, and central C. Thus, for any object A, the functor $F : \mathbf{P}^\bullet \to \mathbf{P}$ given by $F(C) = C \parr A$ has a left adjoint $G : \mathbf{P} \to \mathbf{P}^\bullet$, given by $G(B) = \bot^{A^B}$. The unit of this

adjunction is

$$\mathit{coapp} : B \xrightarrow{\partial} A^{A^B} \xrightarrow{\cong} \bot^{A^B} \parr A.$$

We denote the image of a map $f : B \to C \parr A$ under the adjunction by $\mathit{cocurry}(f) : \bot^{A^B} \to C$. The maps *coapp* and *cocurry* do neither define a co-closed structure on $\mathbf{P}$, nor on $\mathbf{P}^\bullet$. However, from the adjunction, one has



for all $f : B \to C \parr A$, and moreover $\mathit{cocurry}(f)$ is the unique *central* morphism making this diagram commute. Thus, $\bot^{A^B}$ defines a *weak* co-closed structure on $\mathbf{P}$, and we write

$$B \mathrel{\circ\!\!-} A := \bot^{A^B}.$$

4.2. *Co-control categories and $\otimes\neg$-categories*

For the interpretation of the call-by-value $\lambda\mu$-calculus, it will be convenient to dualize the notion of a control category. A ***co-control category*** is the categorical dual of a control category. In particular, it has finite coproducts $A + B$ with initial object 0, co-exponentials which we write as B_A, a pretensor $A \otimes B$ with unit I, and a weak closed structure $A \multimap B$. The following table lists some notation that we are going to use for objects and morphisms of a co-control category:

On objects		On morphisms	
Control	Co-control	Control categories	Co-control categories
1	0	$\Diamond : A \to 1$	$\Box : 0 \to A$
$A \times B$	$A + B$	$i : \bot \to A$	$t : A \to I$
$\bot$	I	$\nabla : A \parr A \to A$	$\Delta : A \to A \otimes A$
$A \parr B$	$A \otimes B$	$\epsilon : B^A \times A \to B$	$\jmath : B \to B_A + A$
B^A	B_A	$f^\star : C \to B^A$	${}^\star f : B_A \to C$
$B \mathrel{\circ\!\!-} A$	$A \multimap B$	$\mathit{coapp} : B \to (B \mathrel{\circ\!\!-} A) \parr A$	$\mathit{app} : (A \multimap B) \otimes A \to B$
		$\mathit{cocurry}(f) : (B \mathrel{\circ\!\!-} A) \to C$	$\mathit{curry}(f) : C \to (A \multimap B)$

We use the usual notation for coproducts. Following Thielecke (1997), the dual of the map θ is called *thunk*, and the dual of ∂ is called *force*. The remaining structural maps keep the same names as their duals.

Every co-control category is a $\otimes\neg$-category in the sense of Thielecke (1997), where $\neg A$ is defined as $I_A \cong A \multimap 0$. Notice that one has $B_A \cong (\neg A) \otimes B$ and $A \multimap B \cong \neg(A \otimes \neg B)$; thus each two of the constructs B_A, $A \multimap B$, and $\neg A$ can be defined in terms of the third. Conversely, one can show that every $\otimes\neg$-category can be fully and faithfully embedded in a co-control category. Thus, co-control categories can be seen as a natural extension of $\otimes\neg$-categories with finite coproducts. The presence of finite coproducts is important for the duality result in Section 8.

Table 1. *The signature of control categories*

Nullary morphism constructors:

id :	$A \to A$
$\Diamond$:	$A \to 1$
π_1 :	$A \times B \to A$
π_2 :	$A \times B \to B$
ϵ :	$B^A \times A \to B$
a :	$(A \parr B) \parr C \to A \parr (B \parr C)$
l :	$A \to A \parr \bot$
c :	$A \parr B \to B \parr A$
i :	$\bot \to A$
∇ :	$A \parr A \to A$
d :	$(A \parr C) \times (B \parr C) \to (A \times B) \parr C$
s^{-1} :	$(B \parr C)^A \to B^A \parr C$

Object constructors:

$$1 \quad \bot \quad A \times B \quad A \to B \quad A \parr B$$

Binary and unary morphism constructors:

$$\frac{f : A \to B \qquad g : B \to C}{g \circ f : A \to C}$$

$$\frac{f : A \to B \qquad g : A \to C}{\langle f, g\rangle : A \to B \times C}$$

$$\frac{f : A \times B \to C}{f^\star : A \to C^B}$$

$$\frac{f : A \to B}{f \parr C : A \parr C \to B \parr C}$$

4.3. *Control categories as algebras*

The structure of control categories, like that of cartesian-closed categories, is equational in the sense of Lambek and Scott (1986). This means that the structure can be given by object constructors, morphism constructors, and universally quantified equations on hom-sets. Any categorical structure that is given in this way enjoys good properties, because the usual algebraic constructions, such as constructing a free algebra or a quotient, have categorical equivalents. Thus, it makes sense to speak of a congruence relation on a control category, to take a quotient, or to freely adjoin a class of arrows.

The object and morphism constructors of control categories are shown in Table 1. Here, it is understood that each given morphism constructor actually stands for a family of constructors, indexed by objects. Some constructors that appear in our definition of control categories are not shown here; they are definable in terms of the remaining ones, and are thus redundant.

The structure of control categories is given by type-indexed equations (with variables) on hom-sets over this signature. These equations can be found in the definitions in Sections 2.1 through 2.4, and we do not repeat them here. For illustration, let us only point out that the requirement that a certain morphism is focal can indeed be expressed equationally. For instance, the focality of $\pi_1 : A \times B \to A$ is expressed by the following three equations:

$$\begin{aligned} \pi_1 \circ i_{A\times B} &= i_A, \\ \pi_1 \circ \nabla_{A\times B} &= \nabla_A \circ (A \parr \pi_1) \circ (\pi_1 \parr (A \times B)), \\ (\pi_1 \parr D) \circ ((A \times B) \parr g) &= (A \parr g) \circ (\pi_1 \parr C), \qquad \text{for all } g : C \to D. \end{aligned}$$

4.4. *The center of a category of continuations*

We have remarked in Section 2.5 that in a category of continuations $R^{\mathbf{C}}$, any morphism of the form $R^g : R^A \to R^B$, for $g : B \to A$ in $\mathbf{C}$, is central. As Führmann has shown, the converse is true iff $\mathbf{C}$ satisfies Moggi's *equalizing requirement* (Moggi 1988; Führmann 1999). We say that an object $A \in |\mathbf{C}|$ satisfies the equalizing requirement if the canonical morphism $\partial_A : A \to R^{R^A}$

is an equalizer in the following diagram:

$$A \xrightarrow{\partial_A} R^{R^A} \underset{R^{R^{\partial_A}}}{\overset{\partial_{R^{R^A}}}{\rightrightarrows}} R^{R^{R^{R^A}}} .$$

We say that $\mathbf{C}$ satisfies the equalizing requirement if all of its objects do.

Remark 4.1. The equalizing requirement is satisfied if ∂_A is a split monic, which is the case, for instance, if $\mathbf{C}$ is the category of non-empty sets and $|R| \geq 2$.

Lemma 4.2. *Let $R^{\mathbf{C}}$ be a category of continuations. For any object $A \in |\mathbf{C}|$, the following are equivalent:*

(1) Every central morphism $f : R^A \to R^B$ is of the form R^g, for some $g : B \to A$.
(2) The object A satisfies the equalizing requirement.

Proof. Denote by $f^\star_\star : B \to R^{R^A}$ the curry and uncurry of a map $f : R^A \to R^B$. Notice that if $g : B \to A$, then $f = R^g$ if and only if $f^\star_\star = \partial_A \circ g$. Also note that such g is necessarily unique, since we have assumed, in the definition of a response category, that ∂_A is monic. By Lemma 3.10, f is central if and only if $f \circ \theta_{R^A} = \theta_{R^B} \circ R^{R^f}$. In a category of continuations, we have $\theta_{R^A} = R^{\partial_A}$, and by ccc manipulations, it follows that f is central if and only if $\partial_{R^{R^A}} \circ f^\star_\star = R^{R^{\partial_A}} \circ f^\star_\star$. It follows that every central morphism $f : R^A \to R^B$ is of the form R^g, for some g, if and only if for every f, it is the case that $\partial_{R^{R^A}} \circ f^\star_\star = R^{R^{\partial_A}} \circ f^\star_\star$ implies $f^\star_\star = \partial_A \circ g$, for some g, if and only if A satisfies the equalizing requirement. □

Remark 4.3. Not every response category satisfies the equalizing requirement. For a trivial counterexample, let $\mathbf{C}$ be the full sub-ccc $\{0, 1\}$ of the category of sets. Let $R = 1$ be the terminal object. Notice that, since $\mathbf{C}$ is a poset, ∂_A is monic for all A. However, $A = 0$ does not satisfy the equalizing requirement. Indeed, the unique map $f : R^0 \to R^1$ is central, but not of the form R^g for $g : 1 \to 0$.

4.5. *On explicitly chosen value categories*

In our definition of control categories, the center is a derived notion: a morphism is central if it satisfies certain equations. Thus, the center is not an explicitly given part of the structure. Computationally, the central morphisms are used to model effect-free computations, or *values*, as we will see in our interpretation of the call-by-value $\lambda\mu$-calculus in Section 7. Some authors, such as Jeffrey (1997), prefer to present premonoidal categories together with an explicitly chosen *value category*, which is a fixed subcategory of the center. In the context of premonoidal categories, there is a clear advantage in working with chosen value categories, because functors that preserve the algebraic structure of premonoidal categories do not in general preserve centrality; but by making values part of the given structure, one can require functors to preserve them. Since, on the other hand, functors of control categories automatically preserve the center, the need for value categories is not so clear in this context. Still, it is possible to accommodate this point of view if desired, and in this section we will show that this leads to a slightly improved statement of the Structure Theorem.

Definition 4.4. Let $\mathbf{P}$ be a control category. A subcategory $\mathbf{V}$ is called a ***value category*** if $|\mathbf{P}| \subseteq \mathbf{V} \subseteq \mathbf{P}^\bullet$, and if $\mathbf{V}$ is closed under all the structural operations of control categories (shown in Table 1), except ϵ and currying. We further require that $\mathbf{V}$ contains the maps $s_{A,B,C}$ and $p_{A,B,C}$, and all maps of the form A^f. If $\mathbf{V}$ is a chosen value category of $\mathbf{P}$, then the morphisms of $\mathbf{V}$ are called ***values***.

Suppose $\mathbf{P}$ and $\mathbf{P}'$ are control categories with respective chosen value categories $\mathbf{V}$ and $\mathbf{V}'$. We say that a (weak) functor of control categories $F : \mathbf{P} \to \mathbf{P}'$ ***preserves values*** if it restricts to a functor from $\mathbf{V}$ to $\mathbf{V}'$, i.e., if $f \in \mathbf{V}$ implies $F(f) \in \mathbf{V}'$. In this case, we also write $F : (\mathbf{P}, \mathbf{V}) \to (\mathbf{P}', \mathbf{V}')$. We say that an equivalence $F : \mathbf{P} \to \mathbf{P}'$, $G : \mathbf{P}' \to \mathbf{P}$ of control categories ***respects values*** if it restricts to an equivalence of the categories $\mathbf{V}$ and $\mathbf{V}'$. This means not only that both F and G preserve values, but also that each component of the natural isomorphisms $G \circ F \cong id_{\mathbf{P}}$ and $F \circ G \cong id_{\mathbf{P}'}$ is a value. If $\mathbf{P}$ and $\mathbf{P}'$ are equivalent in this sense, we also write $(\mathbf{P}, \mathbf{V}) \simeq (\mathbf{P}', \mathbf{V}')$.

If $R^{\mathbf{C}}$ is a category of continuations, let $R^{\mathbf{C}}_v$ be the subcategory consisting of morphisms of the form R^f. Then $R^{\mathbf{C}}_v$ is a value category of $R^{\mathbf{C}}$, and we call it the ***canonical value category***. Note that in Remark 4.3, we gave an example of a category of continuations whose canonical value category was strictly contained in the center. Since the center of any control category is itself a value category, this shows that value categories are not in general unique. However, the following lemma shows that at certain types, the values are uniquely determined.

Lemma 4.5. *If $\mathbf{P}$ is a control category with chosen value category $\mathbf{V}$, then $\mathbf{V}(\perp^A, B) = \mathbf{P}^\bullet(\perp^A, B)$.*

Proof. By the proof of Lemma 3.12, any central map $f : \perp^A \to B$ is of the form $B^g \circ p$, and hence a value. □

Notice that this implies that in a co-control category, the values at the call-by-value function type $A \multimap B = \neg(A_B)$ are uniquely determined. The values $f : C \to (A \multimap B)$ are precisely the maps of the form $f = \mathit{curry}(g)$. Under our interpretation of the $\lambda\mu$-calculus in Section 7, these maps are precisely the lambda abstractions.

In the context of chosen value categories, we can give an improved version of the Structure Theorem. In Theorem 3.18, we have shown that every control category $\mathbf{P}$ is equivalent to a category of continuations $R^{\mathbf{C}}$. However, $\mathbf{C}$ is not in general uniquely determined by $\mathbf{P}$. It turns out that with respect to a chosen value category $\mathbf{V}$ of $\mathbf{P}$, the category $\mathbf{C}$ is unique up to equivalence. This is made precise in the following theorem:

Theorem 4.6 (Second Structure Theorem). *Let $\mathbf{P}$ be a control category with chosen value category $\mathbf{V}$. Then there is a response category $\mathbf{C}$ such that $(\mathbf{P}, \mathbf{V}) \simeq (R^{\mathbf{C}}, R^{\mathbf{C}}_v)$. Moreover, $\mathbf{C}$ is unique up to equivalence of response categories.*

Proof. Existence: Let $\mathbf{C} = \mathbf{V}^{op}$. One proves $\mathbf{P} \simeq R^{\mathbf{C}}$ exactly as in the proof of Theorem 3.18, taking $\mathbf{V}$ in place of $\mathbf{P}^\bullet$. Lemma 4.5 serves to ensure that $\mathbf{V}$ has all the properties of $\mathbf{P}^\bullet$ that were relevant to the proof. Moreover, under the equivalence, the images of values $f \in \mathbf{V}$ are precisely the values $R^{\hat{f}} \in R^{\mathbf{C}}_v$. Uniqueness: First, observe that any response category $\mathbf{D}$ is equivalent to the dual of $R^{\mathbf{D}}_v$ via the contravariant functor that maps A to R^A and f to R^f. Clearly, this functor is full and onto objects; it is also faithful since $\mathbf{D}$ satisfies the mono requirement. Now suppose

Table 2. *The typing rules for the λμ-calculus*

$$(var)\quad \frac{}{\Gamma \vdash x : A \mid \Delta}\ \text{if } x{:}A \in \Gamma \qquad (const)\quad \frac{}{\Gamma \vdash c^A : A \mid \Delta}$$

$$(*)\quad \frac{}{\Gamma \vdash * : \top \mid \Delta} \qquad (pair)\quad \frac{\Gamma \vdash M : A \mid \Delta \qquad \Gamma \vdash N : B \mid \Delta}{\Gamma \vdash \langle M, N\rangle : A \wedge B \mid \Delta}$$

$$(\pi_1)\quad \frac{\Gamma \vdash M : A \wedge B \mid \Delta}{\Gamma \vdash \pi_1 M : A \mid \Delta} \qquad (\pi_2)\quad \frac{\Gamma \vdash M : A \wedge B \mid \Delta}{\Gamma \vdash \pi_2 M : B \mid \Delta}$$

$$(app)\quad \frac{\Gamma \vdash M : A \to B \mid \Delta \qquad \Gamma \vdash N : A \mid \Delta}{\Gamma \vdash MN : B \mid \Delta} \qquad (abs)\quad \frac{\Gamma, x{:}A \vdash M : B \mid \Delta}{\Gamma \vdash \lambda x^A.M : A \to B \mid \Delta}$$

$$(name)\quad \frac{\Gamma \vdash M : A \mid \Delta}{\Gamma \vdash [\alpha]M : \bot \mid \Delta}\ \text{if } \alpha{:}A \in \Delta \qquad (\mu)\quad \frac{\Gamma \vdash M : \bot \mid \alpha{:}A, \Delta}{\Gamma \vdash \mu\alpha^A.M : A \mid \Delta}$$

$$(weaken)\quad \frac{\Gamma \vdash M : A \mid \Delta}{\Gamma' \vdash M : A \mid \Delta'}\ \text{if } \Gamma \subseteq \Gamma', \Delta \subseteq \Delta'$$

$(\mathbf{P}, \mathbf{V}) \simeq (R^{\mathbf{D}}, R_v^{\mathbf{D}})$. Then we have $\mathbf{D} \simeq (R_v^{\mathbf{D}})^{op} \simeq \mathbf{V}^{op} = \mathbf{C}$; moreover, this equivalence identifies the response object $R \in \mathbf{D}$ with $R = \widehat{\perp^{\perp}} \in \mathbf{C}$. Thus, $\mathbf{D}$ and $\mathbf{C}$ are equivalent as response categories. □

5. The λμ-calculus

5.1. *The syntax of the λμ-calculus*

We will show how to interpret Parigot's $\lambda\mu$-calculus (Parigot 1992) in a control category. We begin by reviewing the syntax of the $\lambda\mu$-calculus with finite products. Let $\sigma, \tau, \ldots$ range over a set $\mathcal{B}$ of ***type constants***. Types, ranged over by $A, B, \ldots$, are constructed by the grammar:

$$A ::= \sigma \mid \top \mid A \wedge B \mid A \to B \mid \bot$$

Let $\mathcal{V}$ and $\mathcal{N}$ be two given, infinite, disjoint sets of ***object variables*** $x, y, \ldots$ and ***control variables*** $\alpha, \beta, \ldots$, respectively. Control variables are also called ***names***. Let $\mathcal{K}$ be a set of typed ***object constants*** $c^A, d^B, \ldots$. The pair $(\mathcal{B}, \mathcal{K})$ is called a ***signature of the λμ-calculus***, and sometimes denoted by Σ. Raw terms $M, N, \ldots$ are constructed by the grammar:

$$M ::= x \mid c^A \mid * \mid \langle M, N\rangle \mid \pi_1 M \mid \pi_2 M \mid MN \mid \lambda x^A.M \mid [\alpha]M \mid \mu\alpha^A.M$$

A term of the form $\mu\alpha^A.M$ is called a ***μ-abstraction***, and a term of the form $[\alpha]M$ is called a ***named term***. In the terms $\lambda x^A.M$ and $\mu\alpha^A.M$, the object variable x, respectively the control variable α, is bound. As usual, we identify raw terms up to renaming of bound object and control variables.

The typing of the $\lambda\mu$-calculus is defined as follows. An ***object context*** is a finite, possibly empty sequence $\Gamma = x_1{:}B_1, x_2{:}B_2, \ldots, x_n{:}B_n$ of pairs of an object variable and a type, such that $x_i \neq x_j$ for all $i \neq j$. We write $\Gamma \subseteq \Gamma'$ if Γ is contained in Γ' as a set. A ***control context*** $\Delta = \alpha_1{:}A_1, \alpha_2{:}A_2, \ldots, \alpha_m{:}A_m$ is defined analogously. A ***typing judgment*** is an expression of the form $\Gamma \vdash M : A \mid \Delta$, i.e., a quadruple consisting of an object context, a term, a type, and a control context. In the logical interpretation of a sequent, the symbol "$\vdash$" stands for an implication, and the symbol "$\mid$" stands for a disjunction. ***Valid typing judgments*** are derived by

the rules in Table 2. An ***equation*** is an expression of the form $\Gamma \vdash M = N : A \mid \Delta$, where $\Gamma \vdash M : A \mid \Delta$ and $\Gamma \vdash N : A \mid \Delta$ are valid typing judgments.

5.2. *An informal description of the semantics of the λμ-calculus*

To motivate the constructs of the $\lambda\mu$-calculus, we first give an informal discussion of the intended semantics. A formal semantics will be given by means of CPS translations in Sections 6 and 7.

The μ-abstractions and named terms of the $\lambda\mu$-calculus are operators that influence the sequential flow of control during the evaluation of a term. Informally, when a subterm is evaluated, one of two things could happen: it could return a value to its environment, or it could cause the control flow to jump to some other part of the program. A term that contains such a jump may never return a value to its environment at all, in which case it can be given the type $\bot$, which is the "empty type" of which there are no values.

In the $\lambda\mu$-calculus, a prototypical term of type $\bot$ is the term $[\alpha]M$. It does not return anything, but passes the value of M to a control variable named α instead. One can think of α as a named channel, and of the value of M as being sent along this channel. One also says that M is *thrown* to α. Channels are typed: if α has type A, then this means that values of type A can be thrown to α.

We also need a binding construct for channels. The μ-abstraction $N = \mu\alpha^A.M$ creates a named channel α and then begins to evaluate M. If in the process of evaluating M, some value gets thrown to α, then this value immediately becomes the value of the whole expression N. The evaluation of M is not continued in this case. Thus, since α is declared to be of type A, the term N must have type A as well. What should the type of M be? In the $\lambda\mu$-calculus, the body M of a μ-abstraction has type $\bot$. Thus, the question of what to do when M returns a value does not arise. However, this is not a serious restriction: one can easily deal with terms M of arbitrary type by using the idiom $\mu\alpha^A.[\beta]M$, or even $\mu\alpha^A.[\alpha]M$ when M has type A.

A typing judgment $x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$ means that M is a well-typed term with at most n free typed object variables and at most m free typed control variables. One can think of M as a function in n arguments which has $m + 1$ possible result channels: it may return an ordinary value of type A, or it may return an exceptional value of type A_i on some channel α_i.

On the surface, there is a certain similarity between the control constructs of the $\lambda\mu$-calculus and the exception handling mechanism of ML. As a first approximation, one may think of throwing a value V to α as raising an exception α with value V. Similarly, one may think of the term $\mu\alpha^A.M$ as providing a handler for the exception α. However, this analogy is only superficial, and there is an important difference between ML exception handling and the $\lambda\mu$-calculus: the latter is *statically scoped*. This means, the term $N = \mu\alpha^A.M$ binds those occurrences of α in M in its *syntactic* scope. Occurrences of α that are substituted into M (for instance as the result of reducing a λ-redex) are *not* bound in N. On the other hand, ML exceptions are *dynamically scoped*, which allows a function, among other things, to handle an exception that one of its arguments throws. Because of its static scoping, the $\lambda\mu$-calculus is a calculus of *continuations*, not exceptions, and the μ-abstraction mechanism is closely related to control operators such as `callcc` in Scheme, or Felleisen's $\mathcal{C}$.

5.3. *Adding classical disjunction to the $\lambda\mu$-calculus*

The $\lambda\mu$-calculus can be regarded, and was originally conceived, as a term calculus for multi-conclusioned classical sequent calculus proofs. This connection to logic suggests that one should be able to extend the calculus to include a disjunction type constructor. Indeed, there is a standard way of adding disjoint sum types to the lambda calculus via left and right injections and a case construct. However, the proof theory of disjunction in classical logic is quite different from that in intuitionistic logic, and Pym and Ritter (1998) show how to add a different, classical, disjunction type to the $\lambda\mu$-calculus. They also show that these classical disjunction types are strictly different from the disjoint sum type under the call-by-name semantics. On the other hand, we shall see that under call-by-value, the two disjunction types coincide.

In the following, we essentially adopt Pym and Ritter's classical disjunction type, although we use a slightly different, more symmetric syntax for terms. Formally, we add one type constructor and two term constructors to the $\lambda\mu$-calculus:

$$\begin{array}{rcl} A & ::= & \ldots \mid A \vee B \\ M & ::= & \ldots \mid [\alpha,\beta]M \mid \mu(\alpha^A,\beta^B).M \end{array}$$

The two new term constructors are generalizations of μ-abstraction and naming that deal with two control variables, rather than one. Informally, one may think of a value of type $A \vee B$ as being either a value of type A or a value of type B. Depending on which is the case, the term $[\alpha,\beta]M$ will throw the value of M to α or β. Similarly, the term $\mu(\alpha^A,\beta^B).M$ catches any value that is thrown to α or β, and synthesizes it to a value of type $A \vee B$. The typing rules for disjunction are:

$$(\textit{name}') \quad \frac{\Gamma \vdash M : A \vee B \mid \Delta}{\Gamma \vdash [\alpha,\beta]M : \bot \mid \Delta} \quad \text{if } \alpha{:}A, \beta{:}B \in \Delta,$$

$$(\mu') \quad \frac{\Gamma \vdash M : \bot \mid \alpha{:}A, \beta{:}B, \Delta}{\Gamma \vdash \mu(\alpha^A,\beta^B).M : A \vee B \mid \Delta}.$$

Notice that the typing rules imply that in the pattern $\mu(\alpha^A,\beta^B).M$, the variables α and β are different. When writing terms in the disjunctive $\lambda\mu$-calculus, we will sometimes use a more generous syntax for μ-abstractions and named terms. For instance, $\mu(\alpha^A,(\beta^B,\gamma^C)).M$ is an abbreviation for $\mu(\alpha^A,\delta^{B\vee C}).[\delta]\mu(\beta^B,\gamma^C).M$, and similarly $[\alpha,[\beta,\gamma]]M$ is syntactic sugar for the term $[\beta,\gamma]\mu\delta^{B\vee C}.[\alpha,\delta]M$.

The disjunction $A \vee B$ is *classical*. For instance,

$$M = \mu(\alpha^A,\beta^{A\to\bot}).[\beta]\lambda x^A.[\alpha]x$$

is a closed term of type $A \vee (A \to \bot)$. It is instructive to examine the behavior of this term, because it is an illustration of how the static scoping works. Informally, when the term M is initially evaluated, it will return a closure $\lambda x^A.[\alpha]x$ of type $A \to \bot$ to its environment. Should the environment ever attempt to apply this closure to some value v of type A, then the control flow will jump back to the term M in the environment in which it was originally called. At that point, M will evaluate to v.

6. The call-by-name interpretation of the $\lambda\mu$-calculus

The $\lambda\mu$-calculus was originally introduced as a call-by-name language (Parigot 1992; Ong 1996), although Ong and Steward have later given it a call-by-value interpretation (Ong and Stewart 1997). We will first consider the call-by-name semantics, and leave the call-by-value semantics for the next section.

The operational semantics of the call-by-name $\lambda\mu$-calculus can be given in several different familiar styles. Parigot gave a strongly normalizing system of reductions for his original calculus, an approach that was generalized to the extensional and disjunctive case by Pym and Ritter (1998). However, the reduction rules for the control operators are less than intuitive, and they involve complex substitution operations and permutations of contexts.

For our purposes, it is more convenient to consider a continuation passing style (CPS) semantics. One such semantics, based on Plotkin's original call-by-name semantics for the simply-typed lambda calculus (Plotkin 1975), was given by De Groote (1994a). We adopt a different CPS translation which was given by Hofmann and Streicher (1997) and which takes advantage of a richer target language with finite sums and products. Streicher and Reus (1998) demonstrated that such a CPS translation can serve as the basis for an abstract machine model, yielding a stack-based Krivine machine for the call-by-name $\lambda\mu$-calculus. We extend the CPS translation to include disjunction types, and take it as the basis for our categorical interpretation of the call-by-name disjunctive $\lambda\mu$-calculus in a control category. It is also possible to systematically extend the Krivine machine semantics to account for disjunction types. This is carried out in (Selinger 1998).

6.1. *The call-by-name CPS translation*

Consider the disjunctive $\lambda\mu$-calculus over some signature $(\mathcal{B},\mathcal{K})$. We will give the call-by-name semantics of this calculus by a CPS translation. The target language of the translation is a lambda calculus $\lambda^{R\times+}$ with sum, products, and a distinguished type R of responses. Function types are restricted to the case $A \to R$, and consequently, lambda abstractions $\lambda x.M$ occur only when M has type R. Let $=_{\beta\eta}$ denote the usual $\beta\eta$-equivalence of $\lambda^{R\times+}$, with surjective pairing and exhaustive sums.

To keep the notation brief, we use various forms of syntactic sugar for the sums and products of the target calculus. We use patterned lambda abstraction $\lambda\langle x,y\rangle^{A\times B}.M$, which is customarily defined as an abbreviation for $\lambda z^{A\times B}.M[\pi_1 z/x, \pi_2 z/y]$. We also use the co-pairing notation $[M,N]$ as a shorthand for the expression $(\lambda k^{A+B}.\text{case } k \text{ of inl } k_1 \Rightarrow Mk_1 \mid \text{inr } k_2 \Rightarrow Nk_2)$. Notice that $[M,N]$ is the term that corresponds to $\langle M,N\rangle$ under the canonical isomorphism $(A+B\to R) \cong (A\to R)\times(B\to R)$. We also use lambda abstraction patterns for co-pairing; thus $\lambda[x,y]^{A+B\to R}.M$ is a shorthand for $\lambda z^{A+B\to R}.M[\lambda a^A.z(\text{inl } a)/x, \lambda b^B.z(\text{inr } b)/y]$. The initial type 0 is equipped with a type cast operator: If M has type 0, then $\Box_A M$ has type A. By $*$, we denote the canonical term of the unit type 1.

Definition 6.1 (Call-by-name CPS translation). We assume that the target calculus has a type constant $\tilde{\sigma}$ for each type constant $\sigma \in \mathcal{B}$ of $\lambda\mu$. For each type A of the $\lambda\mu$-calculus, we define a pair of types K_A and C_A of the target calculus, called, respectively, the type of ***continuations***

Table 3. *The CPS translation of the call-by-name $\lambda\mu$-calculus*

$\underline{x}$	$=$	$\lambda k^{K_A}.\tilde{x}k$	where $x : A$
$\underline{c^A}$	$=$	$\lambda k^{K_A}.\tilde{c}k$	
$\underline{*}$	$=$	$\lambda k^{K_\top}.\Box_R k$	
$\underline{\langle M, N\rangle}$	$=$	$\lambda k^{K_{A\wedge B}}.[\underline{M}, \underline{N}]k$	where $M : A, N : B$
$\underline{\pi_1 M}$	$=$	$\lambda k^{K_A}.\underline{M}(\mathrm{inl}\ k)$	where $M : A \wedge B$
$\underline{\pi_2 M}$	$=$	$\lambda k^{K_B}.\underline{M}(\mathrm{inr}\ k)$	where $M : A \wedge B$
$\underline{MN}$	$=$	$\lambda k^{K_B}.\underline{M}\langle \underline{N}, k\rangle$	where $M : A \to B, N : A$
$\underline{\lambda x^A.M}$	$=$	$\lambda\langle \tilde{x}, k\rangle^{K_{A\to B}}.\underline{M}k$	where $M : B$
$\underline{[\alpha]M}$	$=$	$\lambda k^{K_\bot}.\underline{M}\tilde{\alpha}$	where $M : A$
$\underline{\mu\alpha^A.M}$	$=$	$\lambda\tilde{\alpha}^{K_A}.\underline{M}*$	where $M : \bot$
$\underline{[\alpha, \beta]M}$	$=$	$\lambda k^{K_\bot}.\underline{M}\langle\tilde{\alpha}, \tilde{\beta}\rangle$	where $M : A \vee B$
$\underline{\mu(\alpha^A, \beta^B).M}$	$=$	$\lambda\langle\tilde{\alpha}, \tilde{\beta}\rangle^{K_{A\vee B}}.\underline{M}*$	where $M : \bot$

and of ***computations*** of type A:

$$\begin{array}{lcll} K_\sigma & = & \tilde{\sigma}, & \text{where } \sigma \text{ is a type constant,} \\ K_\top & = & 0, & \\ K_{A\wedge B} & = & K_A + K_B, & \\ K_{A\to B} & = & C_A \times K_B, & \\ K_\bot & = & 1, & \\ K_{A\vee B} & = & K_A \times K_B, & \\ C_A & = & K_A \to R. & \end{array}$$

For each object constant $c^A \in \mathcal{K}$ of the $\lambda\mu$-calculus, we assume that the target calculus has a constant $\tilde{c}$ of type C_A. Moreover, for each object variable x and each control variable α of the $\lambda\mu$-calculus, we assume a distinct variable $\tilde{x}$, respectively $\tilde{\alpha}$, of the target calculus. The call-by-name CPS translation $\underline{M}$ of a typed term M is given in Table 3. It respects the typing in the following sense:

$$\frac{x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m}{\tilde{x}_1{:}C_{B_1}, \ldots, \tilde{x}_n{:}C_{B_n}, \tilde{\alpha}_1{:}K_{A_1}, \ldots, \tilde{\alpha}_m{:}K_{A_m} \vdash \underline{M} : C_A}. \tag{1}$$

We also write $\underline{\Gamma \vdash M : A \mid \Delta}$ for the translation of a typing judgment, and similarly for equations.

This particular CPS translation, for the fragment without products and disjunction, was discovered by Hofmann and Streicher (1997). It differs from Plotkin's original call-by-name translation (Plotkin 1975) by introducing one less double negation at function types, thus taking advantage of the richer target language. We compare the two translations in detail in Section 6.5.

Notice that the translation of the control operators is straightforward: they simply exchange current continuations. Thus, the translation reveals the nature of the control variables of the $\lambda\mu$-calculus: they are essentially variables of the target language, to which the user of the source language has limited access. One could take this idea further and allow *arbitrary* expressions e of the target calculus to appear in the construct $[e]M$ of the source calculus. A similar extension was proposed by Streicher and Reus (1998). Such an extension would be in the spirit of Filinski's symmetric lambda calculus (Filinski 1989), as it would put terms and continuations on equal

footing. However, such extensions also lead to an incomprehensible programming style, and since they do not add any expressive power to the language, we do not consider them further.

Definition 6.2. Let M and N be terms of the $\lambda\mu$-calculus such that $\Gamma \vdash M : A \mid \Delta$ and $\Gamma \vdash N : A \mid \Delta$. We say that M and N are ***call-by-name equivalent***, in symbols $M =_n N$, if $\underline{M} =_{\beta\eta} \underline{N}$. More generally, if $\mathcal{T}$ is a theory of the $\lambda^{R\times+}$-calculus, we define the ***call-by-name $\lambda\mu$-theory*** generated by $\mathcal{T}$ to be the set of equations $\{E \mid \underline{E} \in \mathcal{T}\}$.

Note that the class of call-by-name $\lambda\mu$-theories is closed under arbitrary intersections. As a matter of fact, it has a finite equational axiomatization. For the fragment without products and disjunction, this follows from Hofmann and Streicher's result (Hofmann and Streicher 1997). We will show how to obtain a finite axiomatization of the theories of the full calculus as a consequence of the Structure Theorem, after discussing the interpretation of the call-by-name $\lambda\mu$-calculus in a control category.

6.2. *The interpretation of the call-by-name $\lambda\mu$-calculus in a control category*

The target calculus $\lambda^{R\times+}$ of our CPS translation can be interpreted directly in a response category $\mathbf{C}$. Recall that this was a category with distributive finite products and coproducts, a distinguished object R, and exponentials of the form R^A. Let us momentarily identify the types K_A and C_A with their interpretation in $\mathbf{C}$. Then by Property (1), the CPS translation of a typing judgment $x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$ gives rise to a morphism in $\mathbf{C}$:

$$C_{B_1} \times \ldots \times C_{B_n} \times K_{A_1} \times \ldots \times K_{A_m} \to C_A.$$

Using $C_A = R^{K_A}$ and currying, this amounts to a morphism

$$C_{B_1} \times \ldots \times C_{B_n} \to R^{K_A \times K_{A_1} \times \ldots \times K_{A_m}},$$

which lies within the continuation category $R^{\mathbf{C}}$. Thus, we can use the standard premonoidal structure on $R^{\mathbf{C}}$ to write

$$C_{B_1} \times \ldots \times C_{B_n} \to C_A \parr C_{A_1} \parr \ldots \parr C_{A_m}.$$

We have thus eliminated any reference to the continuation types K_A from the interpretation of a typing judgment. Indeed, one can interpret the call-by-name $\lambda\mu$-calculus in a control category $\mathbf{P}$ directly, without explicitly mentioning continuations. The interpretation is very natural:

Definition 6.3 (Categorical call-by-name interpretation). Let $\mathbf{P}$ be a control category. To interpret the $\lambda\mu$-calculus with signature $(\mathcal{B}, \mathcal{K})$, assume a choice of an object $\tilde{\sigma}$ for every type constant $\sigma \in \mathcal{B}$. Each type constructor is interpreted by the corresponding object constructor of control categories:

$$\begin{array}{lcll} [\![\sigma]\!]_n & = & \tilde{\sigma}, & \text{where } \sigma \text{ is a type constant,} \\ [\![\top]\!]_n & = & 1, & \\ [\![A \wedge B]\!]_n & = & [\![A]\!]_n \times [\![B]\!]_n, & \\ [\![A \to B]\!]_n & = & [\![B]\!]_n^{[\![A]\!]_n}, & \\ [\![\bot]\!]_n & = & \bot, & \\ [\![A \vee B]\!]_n & = & [\![A]\!]_n \parr [\![B]\!]_n. & \end{array}$$

Table 4. *The interpretation of the call-by-name $\lambda\mu$-calculus in a control category*

$$
\begin{array}{lcl}
[\![\Gamma \vdash x_i : B_i \mid \Delta]\!]_n & = & \Gamma \xrightarrow{\pi_i} B_i \xrightarrow{w} B_i \parr \Delta \\
[\![\Gamma \vdash c^A : A \mid \Delta]\!]_n & = & \Gamma \xrightarrow{\Diamond} 1 \xrightarrow{\tilde{c}} A \xrightarrow{w} A \parr \Delta \\
[\![\Gamma \vdash * : \top \mid \Delta]\!]_n & = & \Gamma \xrightarrow{\Diamond} 1 \xrightarrow{\cong} 1 \parr \Delta \\
[\![\Gamma \vdash \langle M, N\rangle : A \wedge B \mid \Delta]\!]_n & = & \Gamma \xrightarrow{\langle [\![M]\!]_n, [\![N]\!]_n\rangle} (A \parr \Delta)\times(B \parr \Delta) \xrightarrow{d} (A\times B) \parr \Delta \\
[\![\Gamma \vdash \pi_1 M : A \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} (A\times B) \parr \Delta \xrightarrow{\pi_1 \parr \Delta} A \parr \Delta \\
[\![\Gamma \vdash \pi_2 M : B \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} (A\times B) \parr \Delta \xrightarrow{\pi_2 \parr \Delta} B \parr \Delta \\
[\![\Gamma \vdash MN : B \mid \Delta]\!]_n & = & \Gamma \xrightarrow{\langle [\![M]\!]_n, [\![N]\!]_n\rangle} (B^A \parr \Delta)\times(A \parr \Delta) \xrightarrow{d} (B^A\times A) \parr \Delta \xrightarrow{\epsilon \parr \Delta} B \parr \Delta \\
[\![\Gamma \vdash \lambda x^A.M : A \to B \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]^\star_n} (B \parr \Delta)^A \xrightarrow{s^{-1}} B^A \parr \Delta \\
[\![\Gamma \vdash [\alpha_i]M : \bot \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} A_i \parr \Delta \xrightarrow{w_i \parr \Delta} \Delta \parr \Delta \xrightarrow{\nabla} \Delta \xrightarrow{\cong} \bot \parr \Delta \\
[\![\Gamma \vdash \mu\alpha^A.M : A \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} \bot \parr A \parr \Delta \xrightarrow{\cong} A \parr \Delta \\
[\![\Gamma \vdash [\alpha_i, \alpha_j]M : \bot \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} A_i \parr A_j \parr \Delta \xrightarrow{w_i \parr w_j \parr \Delta} \Delta \parr \Delta \parr \Delta \xrightarrow{\nabla \parr \Delta ; \nabla} \Delta \xrightarrow{\cong} \bot \parr \Delta \\
[\![\Gamma \vdash \mu(\alpha^A, \beta^B).M : A \vee B \mid \Delta]\!]_n & = & \Gamma \xrightarrow{[\![M]\!]_n} \bot \parr A \parr B \parr \Delta \xrightarrow{\cong} (A \parr B) \parr \Delta
\end{array}
$$

If $\Gamma = x_1{:}B_1, \ldots, x_n{:}B_n$ is an object context, we write $[\![\Gamma]\!]_n := [\![B_1]\!]_n \times \ldots \times [\![B_n]\!]_n$, and we denote the ith projection map by $\pi_i : [\![\Gamma]\!]_n \to [\![B_i]\!]_n$. Similarly, if $\Delta = \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$ is a control context, we write $[\![\Delta]\!]_n := [\![A_1]\!]_n \parr \ldots \parr [\![A_m]\!]_n$, and we use the notation $w_j : [\![A_i]\!]_n \to [\![\Delta]\!]_n$ for the jth weakening map. Typing judgments are interpreted relative to a choice of a morphism $\tilde{c} : 1 \to [\![A]\!]_n$ for each object constant $c^A \in \mathcal{K}$. A typing judgment $\Gamma \vdash M : A \mid \Delta$ is interpreted as a morphism

$$[\![\Gamma \vdash M : A \mid \Delta]\!]_n \quad : \quad [\![\Gamma]\!]_n \to [\![A]\!]_n \parr [\![\Delta]\!]_n,$$

which is also abbreviated to $[\![M]\!]_n$. The interpretation is defined by recursion on the structure of M, as shown in Table 4. To keep the notation reasonable, we have omitted the semantic brackets from the interpretation of types, hoping that no confusion will arise.

Lemma 6.4. *If $\mathbf{P} = R^{\mathbf{C}}$ is a category of continuations, then the call-by-name categorical interpretation of the $\lambda\mu$-calculus in $\mathbf{P}$ coincides with the interpretation of the call-by-name CPS translation in $R^{\mathbf{C}}$.* □

From the Lemma, which is easily checked by induction on terms, together with the Structure Theorem 3.18, one immediately gets soundness and completeness for theories:

Proposition 6.5 (Soundness and Completeness). *The theories induced on the $\lambda\mu$-calculus by the call-by-name categorical interpretation are precisely the theories induced by the call-by-name CPS translation.* □

6.3. *The call-by-name $\lambda\mu$-calculus is an internal language for control categories*

Recall from Section 4.3 that the structure of a control category is given by operations on objects and morphisms, and equations on hom-sets. On the image of any given call-by-name interpretation of the $\lambda\mu$-calculus in a control category, all these structural operations, as they appear in Table 1, are in fact definable by operations on types and typing judgments.

Table 5. *Control category operations on typing judgments*

Nullary operations:

$$
\begin{array}{lcll}
id & = & x{:}A & \vdash \; x : A \\
\Diamond & = & x{:}A & \vdash \; * : \top \\
\pi_1 & = & x{:}A \wedge B & \vdash \; \pi_1 x : A \\
\pi_2 & = & x{:}A \wedge B & \vdash \; \pi_2 x : B \\
\epsilon & = & x{:}(A \to B) \wedge A & \vdash \; (\pi_1 x)(\pi_2 x) : B \\
a & = & x{:}(A \vee B) \vee C & \vdash \; \mu(\alpha^A, (\beta^B, \gamma^C)).[[\alpha, \beta], \gamma]x : A \vee (B \vee C) \\
l & = & x{:}A & \vdash \; \mu(\alpha^A, \delta^\bot).[\alpha]x : A \vee \bot \\
c & = & x{:}A \vee B & \vdash \; \mu(\beta^B, \alpha^A).[\alpha, \beta]x : B \vee A \\
i & = & x{:}\bot & \vdash \; \mu\alpha^A.x : A \\
\nabla & = & x{:}A \vee A & \vdash \; \mu\alpha^A.[\alpha, \alpha]x : A \\
d & = & x{:}(A \vee C) \wedge (B \vee C) & \vdash \; \mu(\delta^{A\wedge B}, \gamma^C).[\delta]\langle \mu\alpha^A.[\alpha, \gamma]\pi_1 x, \mu\beta^B.[\beta, \gamma]\pi_2 x\rangle : (A \wedge B) \vee C \\
s^{-1} & = & x{:}A \to (B \vee C) & \vdash \; \mu(\delta^{A\to B}, \gamma^C).[\delta]\lambda z^A.\mu\beta^B.[\beta, \gamma]xz : (A \to B) \vee C
\end{array}
$$

Binary and unary operations:

$$\frac{f = \;\; x{:}A \vdash M : B \qquad g = \;\; x{:}B \vdash N : C}{g \circ f = \;\; x{:}A \vdash (\lambda x^B.N)M : C}$$

$$\frac{f = \;\; x{:}A \vdash M : B \qquad g = \;\; x{:}A \vdash N : C}{\langle f, g\rangle = \;\; x{:}A \vdash \langle M, N\rangle : B \wedge C}$$

$$\frac{f = \;\; x{:}A \wedge B \vdash M : C}{f^\star = \;\; x{:}A \vdash \lambda y^B.(\lambda x^{A\wedge B}.M)\langle x, y\rangle : B \to C}$$

$$\frac{f = \;\; x{:}A \vdash M : B}{f \parr C = \;\; x{:}A \vee C \vdash \mu(\beta^B, \gamma^C).[\beta](\lambda x^A.M)\mu\alpha^A.[\alpha, \gamma]x : B \vee C}$$

Let x be a fixed object variable. We say that a typing judgment is in ***standard form*** if the object context declares exactly the one variable x, and the control context is empty. We abbreviate standard form typing judgments to $x{:}B \vdash M : A$, i.e., we omit the empty object context. Note that every typing judgment $x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$ is equivalent to a standard form

$$x{:}B_1 \wedge \ldots \wedge B_n \vdash \mu(\alpha, \alpha_1, \ldots, \alpha_m).[\alpha](\lambda x_1 \ldots x_n.M)(\pi_1 x) \ldots (\pi_n x) : A \vee A_1 \vee \ldots \vee A_m,$$

in the sense that the two denote the same morphism under any interpretation in a control category. Table 5 defines the syntactic operations on standard form typing judgments which correspond to the structural operations of control categories.

Lemma 6.6. *Under the call-by-name interpretation, the structural operations of control categories are defined by the operations on typing judgments that are shown in Table 5.*

Proof. It is easy to check this case by case. For instance, if the interpretation of $x{:}A \vdash N : B$ is $f : [\![A]\!]_n \to [\![B]\!]_n$ and the interpretation of $x{:}B \vdash M : C$ is $g : [\![B]\!]_n \to [\![C]\!]_n$, then the interpretation of $x{:}A \vdash (\lambda x^B.M)N : C$ is

$$[\![A]\!]_n \xrightarrow{g^\star \times f} [\![C]\!]_n^{[\![B]\!]_n} \times [\![B]\!]_n \xrightarrow{\epsilon} [\![C]\!]_n,$$

which is indeed $g \circ f : [\![A]\!]_n \to [\![C]\!]_n$. □

Definition 6.7 (Syntactic control category). For a given $\lambda\mu$-signature Σ and a call-by-name

theory $\mathcal{T}$, we can construct the ***syntactic control category*** $\mathbf{P}^n_{\Sigma,\mathcal{T}}$ as follows: The objects of $\mathbf{P}^n_{\Sigma,\mathcal{T}}$ are the types of the language, with the object constructors given by the corresponding type constructors. Morphisms from A to B are named by valid standard form typing judgments $x{:}A \vdash M : B$. Two typing judgments $x{:}A \vdash M : B$ and $x{:}A \vdash N : B$ name the same morphism if $(x{:}A \vdash M = N : B) \in \mathcal{T}$. The operations of a control category on morphisms are defined as in Table 5.

Lemma 6.8. *$\mathbf{P}^n_{\Sigma,\mathcal{T}}$ is a well-defined control category.*

Proof. We must show that $\mathbf{P}^n_{\Sigma,\mathcal{T}}$ satisfies all the defining equations of a control category, i.e., that the corresponding equations between typing judgments hold in $\mathcal{T}$. By the fact that $\mathcal{T}$ is a theory, together with the completeness of the categorical interpretation, there is a control category $\mathbf{P}$ together with a call-by-name interpretation $[\![-]\!]_n$, such that $(x{:}A \vdash M = N : B) \in \mathcal{T}$ iff $[\![x{:}A \vdash M : B]\!]_n = [\![x{:}A \vdash N : B]\!]_n$. By Lemma 6.6, the required equations hold in $\mathbf{P}$, thus in $\mathcal{T}$. □

There is a canonical call-by-name interpretation $[\![-]\!]^0_n$ of the $\lambda\mu$-calculus with signature Σ in $\mathbf{P}^n_{\Sigma,\mathcal{T}}$, defined by $\tilde{\sigma} := \sigma$ and $\tilde{c} := x{:}\top \vdash c : A$. It has the property that the interpretation of each typing judgment is call-by-name equivalent to its standard form. The pair $(\mathbf{P}^n_{\Sigma,\mathcal{T}}, [\![-]\!]^0_n)$ is determined up to isomorphism by the following universal property: For each $\mathcal{T}$-respecting call-by-name interpretation $[\![-]\!]_n$ in a control category $\mathbf{P}$, there is a unique strict functor of control categories $F : \mathbf{P}^n_{\Sigma,\mathcal{T}} \to \mathbf{P}$ such that $F[\![A]\!]^0_n = [\![A]\!]_n$ for all types A, and $F[\![\Gamma \vdash M : A \mid \Delta]\!]^0_n = [\![\Gamma \vdash M : A \mid \Delta]\!]_n$ for all valid typing judgments $\Gamma \vdash M : A \mid \Delta$.

The construction of $\mathbf{P}^n_{\Sigma,\mathcal{T}}$ allows us to pass from theories to categories. The opposite is also possible:

Definition 6.9 (The internal language of a control category). Given a small control category $\mathbf{P}$, we can construct from it a signature Σ and a call-by-name theory $\mathcal{T}$ as follows: Σ has the objects of $\mathbf{P}$ as its type constants, and one object constant ${c_f}^{A\to B}$ for each morphism $f : A \to B$. Consider the canonical interpretation of this language in $\mathbf{P}$, namely the one that interprets each type constants by itself and each object constant ${c_f}^{A\to B}$ by $f^\star : 1 \to B^A$. Let $\mathcal{T}$ by the call-by-name theory induced by this interpretation. We call the pair $(\Sigma, \mathcal{T})$ the ***internal call-by-name language*** of $\mathbf{P}$.

Lemma 6.10. *If $(\Sigma, \mathcal{T})$ is the internal call-by-name language of a control category $\mathbf{P}$, then $\mathbf{P}^n_{\Sigma,\mathcal{T}} \simeq \mathbf{P}$.*

Proof. Clearly, the canonical interpretation of the internal language in $\mathbf{P}$ is onto objects and morphisms, as each morphism $f : A \to B$ is the denotation of $x{:}A \vdash c_f x : B$. Thus, the canonical functor of control categories $\mathbf{P}^n_{\Sigma,\mathcal{T}} \to \mathbf{P}$, given by the universal property of $\mathbf{P}^n_{\Sigma,\mathcal{T}}$, is full and onto objects. It is faithful by definition of $\mathcal{T}$. Thus, $\mathbf{P}^n_{\Sigma,\mathcal{T}} \simeq \mathbf{P}$ by Lemma 3.17. □

6.4. *An axiomatization of the call-by-name $\lambda\mu$-theories*

We obtained soundness and completeness of the categorical call-by-name interpretation almost "for free", because of the way theories were defined; namely, in terms of a CPS translation, which

Table 6. *Axioms of the call-by-name $\lambda\mu$-calculus*

Axioms for the lambda calculus with products:				
$(\beta_\to)$	$(\lambda x^A.M)N$	$=$	$M[N/x] : B$	
$(\eta_\to)$	$\lambda x^A.Mx$	$=$	$M : B$	if $x \notin \mathrm{FV}(M)$
$(\beta_\wedge)$	$\pi_i\langle M_1, M_2\rangle$	$=$	$M_i : A_i$	
$(\eta_\wedge)$	$\langle \pi_1 M, \pi_2 M\rangle$	$=$	$M : A \wedge B$	
$(\eta_\top)$	$*$	$=$	$M : \top$	
Axioms for $\lambda\mu$ and disjunction:				
$(\zeta_\to)$	$(\mu\alpha^{A\to B}.M)N$	$=$	$\mu\beta^B.M[[\beta](-)N/[\alpha](-)] : B$	if $\beta \notin \mathrm{FN}(M, N)$
$(\zeta_\wedge)$	$\pi_i(\mu\alpha^{A_1\times A_2}.M)$	$=$	$\mu\beta^{A_i}.M[[\beta]\pi_i(-)/[\alpha](-)] : A_i$	if $\beta \notin \mathrm{FN}(M)$
$(\zeta_\vee)$	$[\alpha, \beta]\mu\gamma^{A\vee B}.M$	$=$	$M[[\alpha, \beta](-)/[\gamma](-)] : \bot$	
(β_μ)	$[\alpha']\mu\alpha^A.M$	$=$	$M[\alpha'/\alpha] : \bot$	
(η_μ)	$\mu\alpha^A.[\alpha]M$	$=$	$M : A$	if $\alpha \notin \mathrm{FN}(M)$
$(\beta_\vee)$	$[\alpha', \beta']\mu(\alpha^A, \beta^B).M$	$=$	$M[\alpha'/\alpha, \beta'/\beta] : \bot$	
$(\eta_\vee)$	$\mu(\alpha^A, \beta^B).[\alpha, \beta]M$	$=$	$M : A \vee B$	if $\alpha, \beta \notin \mathrm{FN}(M)$
$(\beta_\bot)$	$[\xi^\bot]M$	$=$	$M : \bot$	

is already very close to the categorical semantics. Sometimes it is more convenient to have an equational description of theories, for instance, as the basis of a rewrite semantics.

Having shown that the call-by-name $\lambda\mu$-calculus forms an internal language for control categories, we can characterize the call-by-name theories as follows:

Proposition 6.11. *Fix a signature Σ. Then a congruence relation $\mathcal{T}$ on typing judgments is a call-by-name $\lambda\mu$-theory if and only if all of the following hold:*

1. *An equation is in $\mathcal{T}$ if and only if its standard form is in $\mathcal{T}$.*
2. *$\mathbf{P}^n_{\Sigma,\mathcal{T}}$, as constructed in Lemma 6.8, is a well-defined control category.*
3. *The canonical interpretation $[\![-]\!]^0_n : \lambda\mu \to \mathbf{P}^n_{\Sigma,\mathcal{T}}$ interprets each typing judgment by its own standard form.*

Proof. If $\mathcal{T}$ is a theory, then the three conditions hold by the results of the previous section. Conversely, assume the conditions hold. Then $[\![-]\!]^0_n : \lambda\mu \to \mathbf{P}^n_{\Sigma,\mathcal{T}}$ is an interpretation in a control category, and it validates exactly the equations in $\mathcal{T}$. Thus $\mathcal{T}$ is a call-by-name $\lambda\mu$-theory. □

One can use this characterization to give a sound and complete axiom axiomatization of the call-by-name $\lambda\mu$-theories as follows. We write $\mathrm{FV}(M)$, respectively $\mathrm{FN}(M)$, for the free object and control variables of a term M. As before, we identify terms up to α-equivalence, renaming bound variables as necessary to avoid captures. We consider three kinds of substitution. We write $M[N/x]$ for the usual substitution of a term N for an object variable x in M. We write $M[\alpha'/\alpha]$ for the substitution of the context variable α' for the context variable α in M, and $M[\alpha'/\alpha, \beta'/\beta]$ for two such substitutions performed simultaneously. Finally, we consider the so-called ***mixed substitution***: If M is a term, $C(-)$ is a context, and α a name, then the ***mixed substitution*** $M[C(-)/[\alpha](-)]$ is the result of recursively replacing in M any subterm of the form $[\alpha]\,(-)$ by $C(-)$, and any subterm of the form $[\alpha_1, \alpha_2](-)$, where $\alpha \in \{\alpha_1, \alpha_2\}$, by $C(\mu\alpha^A.[\alpha_1, \alpha_2](-))$.

More formally, $M[C(-)/[\alpha](-)]$ is defined by recursion on M. The two important clauses are

$$\begin{array}{lcll} ([\alpha]M)[C(-)/[\alpha](-)] & = & C(M[C(-)/[\alpha](-)]), & \\ ([\alpha_1,\alpha_2]M)[C(-)/[\alpha](-)] & = & C(\mu\alpha^A.[\alpha_1,\alpha_2]M[C(-)/[\alpha](-)]) & \text{if } \alpha \in \{\alpha_1,\alpha_2\}. \end{array}$$

Mixed substitution commutes with all other term forming operations, avoiding captures as necessary.

An axiomatization of the call-by-name $\lambda\mu$-theories is shown in Table 6. To make the axioms more readable, we have not shown the typing contexts explicitly; we assume each equation to be placed in a typing context which makes the left-hand side and right-hand side well-typed. By a congruence relation on terms we mean a set of equations which is reflexive, symmetric, transitive, and closed under the term formation rules (ξ-rules) and under weakening.

Theorem 6.12 (Axiomatization of call-by-name $\lambda\mu$-theories). *Let $\mathcal{T}$ be a set of equations of the disjunctive $\lambda\mu$-calculus over some fixed signature. Then $\mathcal{T}$ is a call-by-name theory if and only if it is a congruence relation on terms that satisfies the equations in Table 6.*

Proof. Soundness is easily verified, for instance via the CPS translation and Proposition 6.5, together with appropriate substitution lemmas. The proof of completeness is a long and tedious verification of the properties in Proposition 6.11. □

Remark 6.13. Pym and Ritter (1998) have given a strongly normalizing, confluent reduction semantics to the call-by-name disjunctive $\lambda\mu$-calculus based on a similar set of axioms, using a slightly different syntax.

6.5. *Comparison with the Plotkin call-by-name translation*

Our call-by-name CPS translation is based on that of Hofmann and Streicher (1997). It differs from Plotkin's original call-by-name CPS translation for the simply-typed lambda calculus (Plotkin 1975) by introducing fewer double negations. To obtain Plotkin's call-by-name translation from ours, change the definition of $K_{A\to B}$ to

$$K_{A\to B} \quad = \quad (C_A \to C_B) \to R.$$

Notice that this is isomorphic to $((C_A \times K_B) \to R) \to R$, and thus to the double negation of our definition of $K_{A\to B}$. One can regard this as a way of working around the absence of products in the target language. In the definition of the CPS translation, one changes the clauses for application and lambda abstraction accordingly:

$$\begin{array}{lcll} \underline{MN} & = & \lambda k^{K_B}.\underline{M}(\lambda m^{C_A\to C_B}.m\underline{N}k) & \text{where } M : A\to B,\ N : A, \\ \underline{\lambda x^A.M} & = & \lambda k^{K_{A\to B}}.k(\lambda x^{C_A}.\underline{M}) & \text{where } M : B. \end{array}$$

This is precisely Plotkin's 1975 call-by-name translation of the simply-typed lambda calculus. Notice that it induces a different semantics than the Hofmann/Streicher translation: for instance, Plotkin's translation does not validate the full η law, whereas the Hofmann/Streicher translation does.

Plotkin's translation, too, can be formulated categorically. The Plotkin call-by-name interpretation $[\![-]\!]_p$ of the $\lambda\mu$-calculus in a control category is defined just like the standard one, except

Table 7. *The CPS translation of the call-by-value $\lambda\mu$-calculus*

$$\begin{array}{lcll} \overline{x} & = & \lambda k^{\mathrm{K}_A}.k\tilde{x} & \text{where } x : A \\ \overline{c^A} & = & \lambda k^{\mathrm{K}_A}.k\tilde{c} & \\ \overline{*} & = & \lambda k^{\mathrm{K}_\top}.k* & \\ \overline{\langle M,N\rangle} & = & \lambda k^{\mathrm{K}_{A\wedge B}}.\overline{M}(\lambda m^{\mathrm{V}_A}.\overline{N}(\lambda n^{\mathrm{V}_B}.k\langle m,n\rangle)) & \text{where } M : A,\ N : B \\ \overline{\pi_1 M} & = & \lambda k^{\mathrm{K}_A}.\overline{M}(\lambda m^{\mathrm{V}_{A\wedge B}}.k\pi_1 m) & \text{where } M : A\wedge B \\ \overline{\pi_2 M} & = & \lambda k^{\mathrm{K}_B}.\overline{M}(\lambda m^{\mathrm{V}_{A\wedge B}}.k\pi_2 m) & \text{where } M : A\wedge B \\ \overline{MN} & = & \lambda k^{\mathrm{K}_B}.\overline{M}(\lambda m^{\mathrm{V}_{A\to B}}.\overline{N}(\lambda n^{\mathrm{V}_A}.m\langle n,k\rangle)) & \text{where } M : A\to B,\ N : A \\ \overline{\lambda x^A.M} & = & \lambda k^{\mathrm{K}_{A\to B}}.k(\lambda\langle\tilde{x},c\rangle^{\mathrm{V}_A\times\mathrm{K}_B}.\overline{M}c) & \text{where } M : B \\ \overline{[\alpha]M} & = & \lambda k^{\mathrm{K}_\bot}.\overline{M}\tilde{\alpha} & \text{where } M : A \\ \overline{\mu\alpha^A.M} & = & \lambda\tilde{\alpha}^{\mathrm{K}_A}.\overline{M}* & \text{where } M : \bot \\ \overline{[\alpha,\beta]M} & = & \lambda k^{\mathrm{K}_\bot}.\overline{M}[\tilde{\alpha},\tilde{\beta}] & \text{where } M : A\vee B \\ \overline{\mu(\alpha^A,\beta^B).M} & = & \lambda[\tilde{\alpha},\tilde{\beta}]^{\mathrm{K}_{A\vee B}}.\overline{M}* & \text{where } M : A \end{array}$$

for the following changes: the interpretation of the function type is changed to

$$[\![A\to B]\!]_p \quad = \quad \bot^{\bot^{[\![B]\!]_p^{[\![A]\!]_p}}},$$

and the interpretation of application and abstraction are changed to

$$\begin{array}{lcl} [\![\Gamma\vdash MN : B \mid \Delta]\!]_p & = & \Gamma \xrightarrow{\langle [\![M]\!]_p,[\![N]\!]_p\rangle} (\bot^{\bot^{B^A}} \parr \Delta)\times(A\parr\Delta) \xrightarrow{(\theta_{B^A}\parr\Delta)\times(A\parr\Delta)} \\ & & (B^A\parr\Delta)\times(A\parr\Delta) \xrightarrow{d} (B^A\times A)\parr\Delta \xrightarrow{\epsilon\parr\Delta} B\parr\Delta, \end{array}$$

$$[\![\Gamma\vdash \lambda x^A.M : A\to B \mid \Delta]\!]_p \quad = \quad \Gamma \xrightarrow{[\![M]\!]_p^\star} (B\parr\Delta)^A \xrightarrow{s^{-1}} B^A\parr\Delta \xrightarrow{\partial_{B^A}\parr\Delta} \bot^{\bot^{B^A}}\parr\Delta.$$

One easily checks that the categorical definition coincides with the syntactic one. Thus, the Plotkin call-by-name semantics can be seen to introduce "one extra thunk" for functional closures.

We remark that the $\lambda\mu$-calculus with Plotkin's call-by-name semantics does *not* form an internal language for control categories; in particular, the interpretation of the object constructors is not sufficient to span the category.

7. The call-by-value interpretation of the $\lambda\mu$-calculus

7.1. *The call-by-value CPS translation*

Using the same target calculus as before, we define a CPS translation for the call-by-value $\lambda\mu$-calculus over a signature $(\mathcal{B},\mathcal{K})$.

Definition 7.1 (Call-by-value CPS translation). As before, we assume that the target calculus has a type constant $\tilde{\sigma}$ for each type constant $\sigma\in\mathcal{B}$ of $\lambda\mu$. For each type A of the $\lambda\mu$-calculus, we define three types V_A, K_A, and C_A of the target calculus, called respectively the type of ***values***,

continuations, and ***computations*** of type A:

$$\begin{array}{lcl} \mathrm{V}_\sigma & = & \tilde{\sigma}, \qquad \text{where } \sigma \text{ is a type constant,} \\ \mathrm{V}_\top & = & 1, \\ \mathrm{V}_{A\wedge B} & = & \mathrm{V}_A \times \mathrm{V}_B, \\ \mathrm{V}_{A\to B} & = & \mathrm{V}_A \times \mathrm{K}_B \to R, \\ \mathrm{V}_\bot & = & 0, \\ \mathrm{V}_{A\vee B} & = & \mathrm{V}_A + \mathrm{V}_B, \\ \mathrm{K}_A & = & \mathrm{V}_A \to R, \\ \mathrm{C}_A & = & \mathrm{K}_A \to R. \end{array}$$

Again, we assume that for each object variable x and control variable α of the $\lambda\mu$-calculus, there is a distinct variable $\tilde{x}$ or $\tilde{\alpha}$ of the target calculus. Further, we assume that the target calculus has a constant $\tilde{c}$ of type V_A for each object constant c^A of $\lambda\mu$. The call-by-value CPS translation $\overline{M}$ of a typed term M is given in Table 7. It respects types in the following sense:

$$\frac{x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m}{\tilde{x}_1{:}\mathrm{V}_{B_1}, \ldots, \tilde{x}_n{:}\mathrm{V}_{B_n}, \tilde{\alpha}_1{:}\mathrm{K}_{A_1}, \ldots, \tilde{\alpha}_m{:}\mathrm{K}_{A_m} \vdash \overline{M} : \mathrm{C}_A}. \tag{2}$$

The difference between the call-by-name and call-by-value interpretations is that in the latter, object variables are interpreted as values, and not as computations. We also write $\overline{\Gamma \vdash M : A \mid \Delta}$ for the translation of a typing judgment, and similarly for equations.

Notice that in the call-by-value CPS translation, the clauses for the control operators are identical to the ones for call-by-name, modulo the identification of $[\alpha, \beta]$ with $\langle \alpha, \beta \rangle$ under the canonical isomorphism $(A + B \to R) \cong (A \to R) \times (B \to R)$. The clauses for the pure lambda calculus part are essentially Plotkin's original ones for call-by-value (Plotkin 1975), except that Plotkin did not use a target calculus with pairs and would have defined $\mathrm{V}_{A\to B} = \mathrm{V}_A \to \mathrm{C}_B$. However, unlike in the call-by-name case, our call-by-value translation coincides with Plotkin's up to isomorphism of types.

As usual, the clauses for pairing and application fix a particular evaluation order for M and N, and in each case, the opposite choice would have been equally plausible.

Definition 7.2. Let M and N be terms of the $\lambda\mu$-calculus such that $\Gamma \vdash M : A \mid \Delta$ and $\Gamma \vdash N : A \mid \Delta$. We say that M and N are ***call-by-value equivalent***, in symbols $M =_v N$, if $\overline{M} =_{\beta\eta} \overline{N}$. More generally, if $\mathcal{T}$ is a theory of the $\lambda^{R\times+}$-calculus, we define the ***call-by-value $\lambda\mu$-theory*** generated by $\mathcal{T}$ to be the set of equations $\{E \mid \overline{E} \in \mathcal{T}\}$.

Remark 7.3. This notion of a call-by-value theory corresponds to Moggi's λ_c-calculus more closely than to Plotkin's λ_v-calculus (Moggi 1988; Plotkin 1975). It is well-known that the λ_c-calculus derives more equations than the λ_v-calculus; for instance, the equation $(\lambda x.x)M = M$ is validated by the λ_c-calculus and by the CPS translation, but it is not derivable in the λ_v-calculus (Moggi 1988, Rem. 4.1). See also the axiomatization in Section 7.4

7.2. *The interpretation of the call-by-value $\lambda\mu$-calculus in a co-control category*

As before, we can interpret the target calculus $\lambda^{R\times+}$ of the CPS translation in a response category $\mathbf{C}$. By Property (2), the interpretation of a typing judgment $x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$ is a morphism

$$\mathrm{V}_{B_1} \times \ldots \times \mathrm{V}_{B_n} \times \mathrm{K}_{A_1} \times \ldots \times \mathrm{K}_{A_m} \to \mathrm{C}_A.$$

By currying, and using $\mathrm{C}_A = R^{\mathrm{K}_A}$, one gets

$$\mathrm{K}_A \times \mathrm{K}_{A_1} \times \ldots \times \mathrm{K}_{A_m} \to R^{\mathrm{V}_{B_1} \times \ldots \times \mathrm{V}_{B_n}},$$

which is a morphism in $R^{\mathbf{C}}$. Using the premonoidal structure, one can rewrite this to

$$\mathrm{K}_A \times \mathrm{K}_{A_1} \times \ldots \times \mathrm{K}_{A_m} \to \mathrm{K}_{B_1} \parr \ldots \parr \mathrm{K}_{B_n}.$$

Thus, we have eliminated the types V_A and C_A from the interpretation. Just as for call-by-name, it is now possible to give a direct categorical interpretation of the call-by-value $\lambda\mu$-calculus in a control category. However, since the arrows go "the wrong way", it is more natural to state the interpretation in terms of co-control categories. Thus, the above typing judgment will be interpreted as a morphism is a co-control category:

$$\mathrm{K}_{B_1} \otimes \ldots \otimes \mathrm{K}_{B_n} \to \mathrm{K}_A + \mathrm{K}_{A_1} + \ldots + \mathrm{K}_{A_m}.$$

Recall from Sections 4.1 and 4.2 the weak closed structure on a co-control category, which is given by the operation $A \multimap B$ and the maps $app : (A \multimap B) \otimes A \to B$ and $curry : (C \otimes A, B) \to (C, A \multimap B)$. This structure is used to interpret the call-by-value function type. Notice that it satisfies some of the laws that one would typically expect for call-by-value function spaces, for instance $I \multimap A \cong (A \multimap 0) \multimap 0) \ncong A$, $A \multimap B \cong (A \otimes (B \multimap 0)) \multimap 0$, $I \cong 0 \multimap 0$, etc.

Definition 7.4 (Categorical interpretation: call-by-value). Let $\mathbf{P}$ be a co-control category. The interpretation of the $\lambda\mu$-calculus with signature $(\mathcal{B}, \mathcal{K})$ proceeds relative to a choice of an object $\tilde{\sigma}$ for every type constant $\sigma \in \mathcal{B}$. Each type constructor is interpreted by the corresponding object

Table 8. *The interpretation of the call-by-value $\lambda\mu$-calculus in a co-control category*

$$\begin{array}{lcl}
[\![\Gamma \vdash x_i : B_i \mid \Delta]\!]_v & = & \Gamma \xrightarrow{w} B_i \xrightarrow{inl} B_i{+}\Delta \\
[\![\Gamma \vdash c : A \mid \Delta]\!]_v & = & \Gamma \xrightarrow{t} I \xrightarrow{\tilde{c}} A \xrightarrow{inl} A{+}\Delta \\
[\![\Gamma \vdash * : \top \mid \Delta]\!]_v & = & \Gamma \xrightarrow{t} I \xrightarrow{inl} I{+}\Delta \\
{}^{\star}[\![\Gamma \vdash \langle M, N\rangle : A \wedge B \mid \Delta]\!]_v & = & \Gamma_\Delta \xrightarrow{\Delta} \Gamma_\Delta \otimes \Gamma_\Delta \xrightarrow{{}^{\star}[\![M]\!]_v \otimes id} A \otimes \Gamma_\Delta \xrightarrow{id \otimes {}^{\star}[\![N]\!]_v} A \otimes B \\
[\![\Gamma \vdash \pi_1 M : A \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} (A \otimes B){+}\Delta \xrightarrow{w+id} A{+}\Delta \\
[\![\Gamma \vdash \pi_2 M : B \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} (A \otimes B){+}\Delta \xrightarrow{w+id} B{+}\Delta \\
{}^{\star}[\![\Gamma \vdash MN : B \mid \Delta]\!]_v & = & \Gamma_\Delta \xrightarrow{\Delta} \Gamma_\Delta \otimes \Gamma_\Delta \xrightarrow{({}^{\star}[\![M]\!]_v \otimes id)\,;\,(id \otimes {}^{\star}[\![N]\!]_v)} (A \multimap B) \otimes A \xrightarrow{app} B \\
{}^{\star}[\![\Gamma \vdash \lambda x^A.M : A \to B \mid \Delta]\!]_v & = & \Gamma_\Delta \xrightarrow{curry(f)} A \multimap B, \text{ where } f = \Gamma_\Delta \otimes A \xrightarrow{\cong} (\Gamma \otimes A)_\Delta \xrightarrow{{}^{\star}[\![M]\!]_v} B \\
[\![\Gamma \vdash [\alpha_i]M : \bot \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} A_i{+}\Delta \xrightarrow{in_i+\Delta} \Delta{+}\Delta \xrightarrow{\nabla} \Delta \xrightarrow{\cong} 0{+}\Delta \\
[\![\Gamma \vdash \mu\alpha^A.M : A \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} 0{+}A{+}\Delta \xrightarrow{\cong} A{+}\Delta \\
[\![\Gamma \vdash [\alpha_i, \alpha_j]M : \bot \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} A_i{+}A_j{+}\Delta \xrightarrow{in_i+in_j+\Delta} \Delta{+}\Delta{+}\Delta \xrightarrow{\nabla+\Delta\,;\,\nabla} \Delta \xrightarrow{\cong} 0{+}\Delta \\
[\![\Gamma \vdash \mu(\alpha^A, \beta^B).M : A \vee B \mid \Delta]\!]_v & = & \Gamma \xrightarrow{[\![M]\!]_v} 0{+}A{+}B{+}\Delta \xrightarrow{\cong} (A{+}B){+}\Delta
\end{array}$$

constructor of co-control categories:

$$\begin{array}{lcl}
[\![\sigma]\!]_v & = & \tilde{\sigma}, \qquad \text{where } \sigma \text{ is a type constant,}\\
[\![\top]\!]_v & = & I,\\
[\![A \wedge B]\!]_v & = & [\![A]\!]_v \otimes [\![B]\!]_v,\\
[\![A \to B]\!]_v & = & [\![A]\!]_v \multimap [\![B]\!]_v,\\
[\![\bot]\!]_v & = & 0,\\
[\![A \vee B]\!]_v & = & [\![A]\!]_v + [\![B]\!]_v.
\end{array}$$

For an object context $\Gamma = x_1{:}B_1, \ldots, x_n{:}B_n$, we write $[\![\Gamma]\!]_v = [\![B_1]\!]_v \otimes \ldots \otimes [\![B_n]\!]_v$, and w_i for the ith weakening map. For a control context $\Delta = \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m$, we write $[\![\Delta]\!]_v = [\![A_1]\!]_v + \ldots + [\![A_m]\!]_v$, and in_j for the jth injection. Typing judgments are interpreted relative to a choice of a *central* morphism $\tilde{c} : I \to [\![A]\!]_v$ for each object constant $c^A \in \mathcal{K}$. The interpretation of a typing judgment $\Gamma \vdash M : A \mid \Delta$ is a morphism

$$[\![\Gamma \vdash M : A \mid \Delta]\!]_v : [\![\Gamma]\!]_v \to [\![A]\!]_v + [\![\Delta]\!]_v,$$

defined by recursion on M as shown in Table 8. In the clauses in Table 8, ${}^\star[\![M]\!]_v : \Gamma_\Delta \to A$ refers to the co-curried form of $[\![M]\!]_v : \Gamma \to A + \Delta$.

Notice how in the clauses for application and pairing, the premonoidal structure forces us to choose an evaluation order. In these clauses, M is evaluated before N.

Lemma 7.5. *If $\mathbf{P} = R^{\mathbf{C}}$ is a category of continuations, then the call-by-value categorical interpretation of the $\lambda\mu$-calculus in $\mathbf{P}^{op}$ coincides with the interpretation of the call-by-value CPS translation in $R^{\mathbf{C}}$.* □

Again, it is easy to check the Lemma by induction on terms. The Structure Theorem 3.18 then immediately yields soundness and completeness for theories:

Proposition 7.6 (Soundness and Completeness). *The theories induced on the $\lambda\mu$-calculus by the call-by-value categorical interpretation are precisely the theories induced by the call-by-value CPS translation.* □

We remark that the use of co-currying in the clauses for pairing, application, and λ-abstraction is essential, even though it looks innocuous. The reader is invited to check, for instance, that $[\![\Gamma \vdash \langle M, N\rangle : A \wedge B \mid \Delta]\!]_v$ is *not* equal to

$$\Gamma \xrightarrow{\Delta} \Gamma \otimes \Gamma \xrightarrow{[\![M]\!]_v \otimes id} (A{+}\Delta) \otimes \Gamma \xrightarrow{id \otimes [\![N]\!]_v} (A{+}\Delta) \otimes (B{+}\Delta) \xrightarrow{f} (A \otimes B) + \Delta,$$

no matter which of the natural maps $f : (A{+}\Delta) \otimes (B{+}\Delta) \to (A \otimes B) + \Delta$ one chooses (there are two such maps). Also notice the use of the co-curried form of the interpretation in the following lemma.

Definition 7.7. A ***value*** of the call-by-value $\lambda\mu$-calculus is a term in the grammar

$$V ::= x \mid c^A \mid * \mid \langle V, V'\rangle \mid \pi_1 V \mid \pi_2 V \mid \lambda x^A.M \mid \mu(\alpha^A, \beta^B).[\alpha]V \mid \mu(\alpha^A, \beta^B).[\beta]V,$$

where in the last two cases, neither α nor β occurs freely in V.

Lemma 7.8. *If V is a value, then ${}^\star[\![\Gamma \vdash V : A \mid \Delta]\!]_v : \Gamma_\Delta \to A$ is central.*

Table 9. *Co-control category operations on typing judgments*

Nullary operations:

id	=	$x{:}A$	$\vdash$	$x : A$
□	=	$x{:}\bot$	$\vdash$	$\mu\alpha^A.x : A$
inl	=	$x{:}A$	$\vdash$	$\mu(\alpha^A, \beta^B).[\alpha]x : A \vee B$
inr	=	$x{:}B$	$\vdash$	$\mu(\alpha^A, \beta^B).[\beta]x : A \vee B$
ɔ	=	$x{:}B$	$\vdash$	$\mu(\gamma^{(A\to\bot)\wedge B}, \alpha^A).[\gamma]\langle \lambda y^A.[\alpha]y, x\rangle : ((A \to \bot) \wedge B) \vee A$
a	=	$x{:}A \wedge (B \wedge C)$	$\vdash$	$\langle\langle \pi_1 x, \pi_1\pi_2 x\rangle, \pi_2\pi_2 x\rangle : (A \wedge B) \wedge C$
l	=	$x{:}A \wedge \top$	$\vdash$	$\pi_1 x : A$
c	=	$x{:}B \wedge A$	$\vdash$	$\langle \pi_2 x, \pi_1 x\rangle : A \wedge B$
t	=	$x{:}A$	$\vdash$	$* : \top$
Δ	=	$x{:}A$	$\vdash$	$\langle x, x\rangle : A \wedge A$
d	=	$x{:}(A \vee B) \wedge C$	$\vdash$	$\mu(\delta^{A\wedge C}, \eta^{B\wedge C}).[\delta]\langle \mu\alpha^A.[\eta]\langle \mu\beta^B.[\alpha, \beta]\pi_1 x, \pi_2 x\rangle, \pi_2 x\rangle : (A \wedge C) \vee (B \wedge C)$
s^{-1}	=	$x{:}((A \to \bot) \wedge B) \wedge C$	$\vdash$	$\langle \pi_1\pi_1 x, \langle \pi_2\pi_1 x, \pi_2 x\rangle\rangle : (A \to \bot) \wedge (B \wedge C)$

Binary and unary operations:

$$\frac{f = \ x{:}A \vdash M : B \qquad g = \ x{:}B \vdash N : C}{g \circ f = \ x{:}A \vdash (\lambda x^B.N)M : C}$$

$$\frac{f = \ x{:}B \vdash M : A \qquad g = \ x{:}C \vdash N : A}{[f, g] = \ x{:}B \vee C \vdash \mu\alpha^A.[\alpha](\lambda x^B.M)(\mu\beta^B.[\alpha](\lambda x^C.N)(\mu\gamma^C.[\beta, \gamma]x)) : A}$$

$$\frac{f = \ x{:}B \vdash M : C \vee A}{{}^\star f = \ x{:}(A \to \bot) \wedge B \vdash \mu\gamma^C.(\pi_1 x)(\mu\alpha^A.[\gamma, \alpha](\lambda x^B.M)(\pi_2 x)) : C}$$

$$\frac{f = \ x{:}A \vdash M : B}{f \otimes C = \ x{:}A \wedge C \vdash \langle(\lambda x^A.M)(\pi_1 x), \pi_2 x\rangle : B \wedge C}$$

Proof. Recall from Section 4.1 that *curry*(f) is always central. This settles the case where V is a lambda abstraction. The other cases are equally obvious. □

Remark 7.9. Since we are interested in equational theories, and not in reduction semantics, we are more liberal with the definition of a value than one would otherwise be. Our notion of value corresponds to the existence predicate of the λ_c-calculus in (Moggi 1988), and it includes terms, such as $\pi_1\langle V, V'\rangle$, that are not in normal form.

7.3. *The call-by-value $\lambda\mu$-calculus is an internal language for co-control categories*

The results in this section are analogous to those for the call-by-name calculus in Section 6.3. Just as we were able to define the structural operations of a control category syntactically by operations on typing judgments under the call-by-name interpretation, we can do the same for the structural operations of a co-control category under the call-by-value interpretation. The operations on typing judgments are shown in Table 9.

Lemma 7.10. *The operations on typing judgments in Table 9 define the corresponding structural operations on a co-control category under the call-by-value interpretation, up to natural isomorphism of objects.* □

Definition 7.11 (Syntactic co-control category). For a $\lambda\mu$-signature Σ and a call-by-value theory $\mathcal{T}$, we construct the ***syntactic co-control category*** $\mathbf{P}^v_{\Sigma,\mathcal{T}}$ as follows: The objects of $\mathbf{P}^v_{\Sigma,\mathcal{T}}$ are

the types of the language. The object constructors are given by the corresponding type constructors, where B_A is defined as $(A \to \bot) \wedge B$. Morphisms from A to B are named by valid standard form typing judgments $x{:}A \vdash M : B$. Two typing judgments $x{:}A \vdash M : B$ and $x{:}A \vdash N : B$ name the same morphism if $(x{:}A \vdash M = N : B) \in \mathcal{T}$. The operations of a co-control category on morphisms are defined as in Table 9.

Lemma 7.12. *$\mathbf{P}^v_{\Sigma,\mathcal{T}}$ is a well-defined co-control category.* □

The canonical call-by-value interpretation $[\![-]\!]^0_v$ of the $\lambda\mu$-calculus with signature Σ in $\mathbf{P}^v_{\Sigma,\mathcal{T}}$ is defined by $\tilde{\sigma} := \sigma$ and $\tilde{c} := x{:}\top \vdash c : A$. The interpretation of each typing judgment is call-by-value equivalent to its standard form. The pair $(\mathbf{P}^v_{\Sigma,\mathcal{T}}, [\![-]\!]^0_v)$ is determined up to isomorphism by the universal property: For each $\mathcal{T}$-respecting call-by-value interpretation $[\![-]\!]_v$ in a co-control category $\mathbf{P}$, there is a unique strict functor of co-control categories $F : \mathbf{P}^v_{\Sigma,\mathcal{T}} \to \mathbf{P}$ such that $F[\![A]\!]^0_v = [\![A]\!]_v$ for all A and $F[\![\Gamma \vdash M : A \mid \Delta]\!]^0_v = [\![\Gamma \vdash M : A \mid \Delta]\!]_v$ for all valid typing judgments $\Gamma \vdash M : A \mid \Delta$.

Definition 7.13 (The internal language of a co-control category). Given a small co-control category $\mathbf{P}$, we construct from it a signature Σ and a call-by-name theory $\mathcal{T}$. The type constants are again the objects of $\mathbf{P}$, and we take one object constant ${c_f}^{A\to B}$ for each morphism $f : A \to B$. Consider the canonical interpretation of this language in $\mathbf{P}$ that interprets type constants by themselves and object constants ${c_f}^{A\to B}$ by $\mathit{curry}(f) : I \to (A \multimap B)$. Recall from Section 4 that $\mathit{curry}(f)$ is always central, and thus this interpretation is well-defined. Let $\mathcal{T}$ by the induced call-by-value theory. The pair $(\Sigma, \mathcal{T})$ is called the ***internal call-by-value language*** of $\mathbf{P}$.

Lemma 7.14. *If $(\Sigma, \mathcal{T})$ is the internal call-by-value language of a co-control category $\mathbf{P}$, then $\mathbf{P}^v_{\Sigma,\mathcal{T}} \simeq \mathbf{P}$.*

Proof. Each morphism $f : A \to B$ is the denotation of $x{:}A \vdash {c_f}x : B$. Thus, the canonical interpretation of the internal language in $\mathbf{P}$ is onto objects and morphisms, and hence the canonical functor of co-control categories $\mathbf{P}^v_{\Sigma,\mathcal{T}} \to \mathbf{P}$, which exists by the universal property, is full and onto objects. It is faithful by definition of $\mathcal{T}$. Thus, $\mathbf{P}^v_{\Sigma,\mathcal{T}} \simeq \mathbf{P}$ by Lemma 3.17. □

7.4. *An axiomatization of the call-by-value $\lambda\mu$-theories*

An analogue of Proposition 6.11 holds for call-by-value theories:

Proposition 7.15. *Fix a signature Σ. Then a congruence relation $\mathcal{T}$ on typing judgments is a call-by-value $\lambda\mu$-theory if and only if all of the following hold:*

1. *An equation is in $\mathcal{T}$ if and only if its standard form is in $\mathcal{T}$.*
2. *$\mathbf{P}^v_{\Sigma,\mathcal{T}}$, as constructed in Lemma 7.12, is a well-defined co-control category.*
3. *For every object constant c, the morphism $x{:}\top \vdash c : A$ is central in $\mathbf{P}^v_{\Sigma,\mathcal{T}}$.*
4. *The canonical interpretation $[\![-]\!]^0_v : \lambda\mu \to \mathbf{P}^v_{\Sigma,\mathcal{T}}$ interprets each typing judgment by its own standard form, up to natural isomorphism of types.* □

As in the call-by-name case, we use this characterization to give a complete axiomatization of the call-by-value $\lambda\mu$-theories. The axioms are shown in Table 10. As before, we have omitted

Table 10. *Axioms of the call-by-value $\lambda\mu$-calculus*

Axioms for the lambda calculus with products:				
$(\beta_\to)$	let $x^A = V$ in M	$=$	$M[V/x] : B$	
$(\eta_\to)$	$\lambda x^A.Vx$	$=$	$V : B$	if $x \notin \mathrm{FV}(V)$
$(\beta_\wedge)$	$\pi_i\langle V_1, V_2\rangle$	$=$	$V_i : A_i$	
$(\eta_\wedge)$	$\langle \pi_1 V, \pi_2 V\rangle$	$=$	$V : A \wedge B$	
$(\eta_\top)$	$*$	$=$	$V : \top$	
(id)	let $x^A = M$ in x	$=$	$M : A$	
$(comp)$	let $y^B = (\text{let } x^A = M \text{ in } N)$ in P	$=$	let $x^A = M$ in let $y^B = N$ in $P : C$	if $x \notin \mathrm{FV}(P)$
(let_{app})	MN	$=$	let $x^{A\to B} = M$ in let $y^A = N$ in $xy : B$	if $x \notin \mathrm{FV}(N)$
(let_{pair})	$\langle M, N\rangle$	$=$	let $x^A = M$ in let $y^B = N$ in $\langle x, y\rangle : A \wedge B$	if $x \notin \mathrm{FV}(N)$
(let_π)	$\pi_i M$	$=$	let $x^{A_1\wedge A_2} = M$ in $\pi_i x : A_i$	
Axioms for $\lambda\mu$ and disjunction:				
(ζ)	let $x^A = \mu\alpha^A.M$ in N	$=$	$\mu\beta^B.M[\text{let } x^A = (-) \text{ in } [\beta]N/[\alpha](-)] : B$	if $\beta \notin \mathrm{FN}(M, N)$
(β_μ)	$[\alpha']\mu\alpha^A.M$	$=$	$M[\alpha'/\alpha] : \bot$	
(η_μ)	$\mu\alpha^A.[\alpha]M$	$=$	$M : A$	if $\alpha \notin \mathrm{FN}(M)$
$(\beta_\vee)$	$[\alpha', \beta']\mu(\alpha^A, \beta^B).M$	$=$	$M[\alpha'/\alpha, \beta'/\beta] : \bot$	
$(\eta_\vee)$	$\mu(\alpha^A, \beta^B).[\alpha, \beta]M$	$=$	$M : A \vee B$	if $\alpha, \beta \notin \mathrm{FN}(M)$
$(\beta_\bot)$	$[\xi^\bot]M$	$=$	$M : \bot$	
(let_{name})	$[\alpha]M$	$=$	let $x^A = M$ in $[\alpha]x : A$	
$(let_{name'})$	$[\alpha, \beta]M$	$=$	let $x^A = M$ in $[\alpha, \beta]x : A$	

the typing contexts. We also use the customary notation $(\text{let } x^A = M \text{ in } N)$ to denote the term $(\lambda x^A.N)M$. The letters V, V_1, and V_2 denote values, as defined in Definition 7.7.

Theorem 7.16 (Axiomatization of call-by-value $\lambda\mu$-theories). *Let $\mathcal{T}$ be a set of equations of the disjunctive $\lambda\mu$-calculus over some fixed signature. Then $\mathcal{T}$ is a call-by-value theory if and only if it is a congruence relation on terms, satisfying the equations in Table 10.*

Proof. Soundness is again easy. Completeness is proved by verifying the conditions of Proposition 7.15. □

Remark 7.17. The above axiomatization combines Moggi's axioms for the computational lambda calculus, some of Ong and Stewart's axioms for the call-by-value $\lambda\mu$-calculus, and the obvious axioms for disjunction. The reader will easily verify that certain other axioms, which are not included in our list, are derivable from it, for instance the following two equations, which each assert the emptiness of the type $\bot$:

$(empty)$	$\Gamma, x{:}\bot$	$\vdash$	M	$=$	$N : A$	$\mid$	Δ	
$(\bot)$	Γ	$\vdash$	$(\text{let } x^\bot = M \text{ in } N)$	$=$	$\mu\alpha^A.M : A$	$\mid$	Δ	if $\alpha \notin \mathrm{FV}(M)$

8. Filinski duality for the $\lambda\mu$-calculus

We have shown that the call-by-name $\lambda\mu$-calculus is an internal language for control categories, and the call-by-value $\lambda\mu$-calculus is an internal language for co-control categories. An immediate but surprising consequence is that the call-by-name and call-by-value calculi are syntactically

Table 11. *The syntactic translation from call-by-value to call-by-name*

On types:

$$
\begin{array}{lcl}
(\!|\sigma|\!) & = & \sigma, \quad \text{where } \sigma \text{ is a type constant} \\
(\!|\top|\!) & = & \bot \\
(\!|A \wedge B|\!) & = & (\!|A|\!) \vee (\!|B|\!) \\
(\!|\bot|\!) & = & \top \\
(\!|A \vee B|\!) & = & (\!|A|\!) \wedge (\!|B|\!) \\
(\!|A \to B|\!) & = & ((\!|B|\!) \to (\!|A|\!)) \to \bot
\end{array}
$$

On terms:

$$
\begin{array}{lcl}
(\!|x|\!) & = & \lambda\kappa^{(\!|A|\!)}.[x]\kappa \\
(\!|c^A|\!) & = & \lambda\kappa^{(\!|A|\!)}.[c^{(\!|A|\!)}]\kappa \\
(\!|*|\!) & = & \lambda\kappa^{(\!|\top|\!)}.\kappa \\
(\!|\langle M, N\rangle|\!) & = & \lambda\kappa^{(\!|A\wedge B|\!)}.(\!|M|\!)(\mu x^{(\!|A|\!)}.(\!|N|\!)(\mu y^{(\!|B|\!)}.[x,y]\kappa)) \\
(\!|\pi_1 M|\!) & = & \lambda\kappa^{(\!|A|\!)}.(\!|M|\!)(\mu(x^{(\!|A|\!)}, y^{(\!|B|\!)}).[x]\kappa) \\
(\!|\pi_2 M|\!) & = & \lambda\kappa^{(\!|B|\!)}.(\!|M|\!)(\mu(x^{(\!|A|\!)}, y^{(\!|B|\!)}).[y]\kappa) \\
(\!|\lambda x^A.M|\!) & = & \lambda\kappa^{(\!|A\to B|\!)}.\kappa(\lambda\beta^B.\mu x^A.(\!|M|\!)\beta) \\
(\!|MN|\!) & = & \lambda\kappa^{(\!|B|\!)}.(\!|M|\!)(\lambda\gamma^{(\!|B|\!)\to(\!|A|\!)}.(\!|N|\!)(\gamma\kappa)) \\
(\!|[\alpha]M|\!) & = & \lambda\kappa^{(\!|\bot|\!)}.(\!|M|\!)\alpha \\
(\!|[\varrho^A]M|\!) & = & \lambda\kappa^{(\!|\bot|\!)}.(\!|M|\!)\varrho^{(\!|A|\!)} \\
(\!|\mu\alpha^A.M|\!) & = & \lambda\kappa^{(\!|A|\!)}.(\lambda\alpha^{(\!|A|\!)}.(\!|M|\!)*)\kappa \\
(\!|[\alpha,\beta]M|\!) & = & \lambda\kappa^{(\!|\bot|\!)}.(\!|M|\!)\langle\alpha,\beta\rangle \\
(\!|\mu(\alpha^A,\beta^B).M|\!) & = & \lambda\kappa^{(\!|A\vee B|\!)}.(\lambda\alpha^A.\lambda\beta^B.(\!|M|\!)*)(\pi_1\kappa)(\pi_2\kappa)
\end{array}
$$

isomorphic to each other. More precisely, there are syntactic translations from call-by-value to call-by-name and vice versa, which are mutually inverse up to natural isomorphism of types and equivalence of terms.

Such a duality between call-by-value and call-by-name equational theories was first discovered by Filinski (1989) in his work on the symmetric lambda calculus. Filinski's calculus treats continuations as first-class objects and it has a special syntax that stresses the symmetry between continuations and values. Unlike Filinski, we are not working with a custom-made language. However, the categorical semantics reveals a close connection: it is not difficult to see that Filinski's symmetric lambda calculus forms another internal language for control categories, and thus that its expressive power equals that of the disjunctive $\lambda\mu$-calculus. Thus, the categorical semantics provides a unified framework in which such dualities can be explained in a way that is independent of any particular syntax.

Computationally, the duality between call-by-name and call-by-value can be understood as a duality between demand-driven and data-driven computation, which reverses the direction of data. Proof-theoretically, it is an extension of De Morgan duality from formulas to proofs.

Formally, the translation of a call-by-name language $(\Sigma, \mathcal{T})$ into a call-by-value language can be achieved by forming the syntactic control category $\mathbf{P}^n_{\Sigma,\mathcal{T}}$, and then considering the internal call-by-value language of $(\mathbf{P}^n_{\Sigma,\mathcal{T}})^{op}$. Similarly, one gets a translation from call-by-value to call-by-name. However these translations are not optimal, because they introduce a lot of unnecessary constants.

It is possible to optimize the translations in such a way that no additional constants are introduced. To do this, we need to extend the syntax of the $\lambda\mu$-calculus just slightly and allow a set

Table 12. *The syntactic translation from call-by-name to call-by-value*

On types:

$$
\begin{array}{lcl}
\langle\!|\sigma|\!\rangle & = & \sigma, \quad \text{where } \sigma \text{ is a type constant} \\
\langle\!|\top|\!\rangle & = & \bot \\
\langle\!|A \wedge B|\!\rangle & = & \langle\!|A|\!\rangle \vee \langle\!|B|\!\rangle \\
\langle\!|\bot|\!\rangle & = & \top \\
\langle\!|A \vee B|\!\rangle & = & \langle\!|A|\!\rangle \wedge \langle\!|B|\!\rangle \\
\langle\!|A \to B|\!\rangle & = & (\langle\!|A|\!\rangle \to \bot) \wedge \langle\!|B|\!\rangle
\end{array}
$$

On terms:

$$
\begin{array}{lcl}
\langle\!|x|\!\rangle & = & \lambda\kappa^{\langle\!|A|\!\rangle}.[x]\kappa \\
\langle\!|c^A|\!\rangle & = & \lambda\kappa^{\langle\!|A|\!\rangle}.[c^{\langle\!|A|\!\rangle}]\kappa \\
\langle\!|*|\!\rangle & = & \lambda\kappa^{\langle\!|\top|\!\rangle}.\kappa \\
\langle\!|\langle M, N\rangle|\!\rangle & = & \lambda\kappa^{\langle\!|A\wedge B|\!\rangle}.\langle\!|M|\!\rangle(\mu x^{\langle\!|A|\!\rangle}.\langle\!|N|\!\rangle(\mu y^{\langle\!|B|\!\rangle}.[x,y]\kappa)) \\
\langle\!|\pi_1 M|\!\rangle & = & \lambda\kappa^{\langle\!|A|\!\rangle}.\langle\!|M|\!\rangle(\mu(x^{\langle\!|A|\!\rangle}, y^{\langle\!|B|\!\rangle}).[x]\kappa) \\
\langle\!|\pi_2 M|\!\rangle & = & \lambda\kappa^{\langle\!|B|\!\rangle}.\langle\!|M|\!\rangle(\mu(x^{\langle\!|A|\!\rangle}, y^{\langle\!|B|\!\rangle}).[y]\kappa) \\
\langle\!|\lambda x^A.M|\!\rangle & = & \lambda\kappa^{\langle\!|A\to B|\!\rangle}.(\pi_1\kappa)(\mu x^{\langle\!|A|\!\rangle}.\langle\!|M|\!\rangle(\pi_2\kappa)) \\
\langle\!|MN|\!\rangle & = & \lambda\kappa^{\langle\!|B|\!\rangle}.\langle\!|M|\!\rangle\langle\langle\!|N|\!\rangle, \kappa\rangle \\
\langle\!|[\alpha]M|\!\rangle & = & \lambda\kappa^{\langle\!|\bot|\!\rangle}.\langle\!|M|\!\rangle\alpha \\
\langle\!|[\varrho^A]M|\!\rangle & = & \lambda\kappa^{\langle\!|\bot|\!\rangle}.\langle\!|M|\!\rangle\varrho^{\langle\!|A|\!\rangle} \\
\langle\!|\mu\alpha^A.M|\!\rangle & = & \lambda\kappa^{\langle\!|A|\!\rangle}.(\lambda\alpha^{\langle\!|A|\!\rangle}.\langle\!|M|\!\rangle*)\kappa \\
\langle\!|[\alpha,\beta]M|\!\rangle & = & \lambda\kappa^{\langle\!|\bot|\!\rangle}.\langle\!|M|\!\rangle\langle\alpha,\beta\rangle \\
\langle\!|\mu(\alpha^A,\beta^B).M|\!\rangle & = & \lambda\kappa^{\langle\!|A\vee B|\!\rangle}.(\lambda\alpha^A.\lambda\beta^B.\langle\!|M|\!\rangle*)(\pi_1\kappa)(\pi_2\kappa)
\end{array}
$$

$\mathcal{K}'$ of typed ***control constants***, in addition to the usual object constants. Thus, a signature for the extended language is a triple $(\mathcal{B}, \mathcal{K}, \mathcal{K}')$. We extend the definition of named terms to the case $[\varrho^A]M$, where ϱ^A is a control constant. The semantics generalizes effortlessly to this extension.

The translation between the call-by-value and call-by-name calculi exchanges object and control constants. It also exchanges object and control variables, and it reverses typing judgments, turning terms "inside out". Thus, a call-by-value function of n arguments with m possible return addresses gets translated into a call-by-name function of m arguments with n return addresses. More precisely, the translations preserves typing in the following sense:

$$\frac{x_1{:}B_1, \ldots, x_n{:}B_n \vdash M : A \mid \alpha_1{:}A_1, \ldots, \alpha_m{:}A_m}{\alpha_1{:}(\!|A_1|\!), \ldots, \alpha_m{:}(\!|A_m|\!), \bullet{:}A \vdash (\!|M|\!)\bullet : \bot \mid x_1{:}(\!|B_1|\!), \ldots, x_n{:}(\!|B_n|\!)}.$$

Because terms are turned "inside out", a special variable $\bullet$ appears in the translation that represents the "outside" of a term. The variable $\bullet$ plays a similar role as the current continuation in a CPS transform. The two translations are shown in Tables 11 and 12. Notice that they are identical, except for the translations of function types, lambda abstraction, and application.

Proposition 8.1. *Both translations preserve CPS transforms, and thus the categorical semantics, up to natural isomorphism of types. It follows that the two translations are mutually inverse, in the sense that*

$$M =_n \mu\alpha.(\!|\langle\!|M|\!\rangle\alpha|\!)* \qquad \textit{and} \qquad M =_v \mu\alpha.\langle\!|(\!|M|\!)\alpha\rangle\!|*,$$

up to natural isomorphisms of types. □

Note that, because the translations preserve CPS transforms, a term and its translation evaluate

in precisely the same manner. Thus, the translations do not just preserve equational theories, but in fact, the operational semantics as well. In particular, given appropriate notions of observation, they also preserve *observational* equivalence.

Remark 8.2. The fact that the tensor product in a co-control category is premonoidal, and not monoidal, is reflected by the well-known fact that in the call-by-value calculus, the following two terms are not equivalent in the presence of side-effects (Thielecke 1997):

$$\begin{array}{l}\text{let } x^A = M \text{ in let } y^B = N \text{ in } P,\\ \text{let } y^B = N \text{ in let } x^A = M \text{ in } P.\end{array}$$

On the other hand, in call-by-name, these two terms are equivalent. The dual of this phenomenon is given by the following two terms, which are equivalent in call-by-value, but not in call-by-name.

$$\begin{array}{l}\text{let } x^A = \mu\alpha^A.(\text{let } y^B = \mu\beta^B.P \text{ in } N) \text{ in } M,\\ \text{let } y^B = \mu\beta^B.(\text{let } x^A = \mu\alpha^A.P \text{ in } M) \text{ in } N.\end{array}$$

Appendix A. Some proofs from Section 3

In this Appendix, we give some technical proofs that were omitted from Section 3. These are included for completeness and reference, and need not be consumed in the first reading. Most of these proofs are diagram chases that are more easily done by hand than typeset.

For any objects A, B, and C, let $wd_{A,B,C} : (A \parr B) \times C \to (A \times C) \parr B$ be the map given by

$$wd_{A,B,C} \quad = \quad (A \parr B) \times C \xrightarrow{(A \parr B) \times w} (A \parr B) \times (C \parr B) \xrightarrow{d} (A \times C) \parr B.$$

Then $wd_{A,B,C}$ is natural in A and C, and natural in discardable B, because w and d are. Moreover, wd satisfies coherence:

$$\begin{array}{ccc}(A \parr B \parr C) \times D & & \\ \downarrow{\scriptstyle wd} & \searrow^{wd} & \\ ((A \parr B) \times D) \parr C & \xrightarrow{wd \parr C} & (A \times D) \parr B \parr C,\end{array} \qquad \begin{array}{ccc}A \times B & & \\ \downarrow{\scriptstyle l \times B} & \searrow^{l} & \\ (A \parr \bot) \times B & \xrightarrow{wd} & (A \times B) \parr \bot,\end{array}$$

$$\begin{array}{ccc}(A \parr B) \times C \times D & & \\ \downarrow{\scriptstyle wd \times D} & \searrow^{wd} & \\ ((A \times C) \parr B) \times D & \xrightarrow{wd} & (A \times C \times D) \parr B.\end{array}$$

These follow from naturality and coherence of *d* and *w*, which in turns follow immediately from their respective definitions. Symmetrically, define

$$\begin{array}{rcl}wd'_{A,B,C} & = & A \times (B \parr C) \xrightarrow{w \times (B \parr C)} (B \parr A) \times (B \parr C) \xrightarrow{d} B \parr (A \times C),\\ wd''_{A,B,C} & = & (A \parr B) \times C \xrightarrow{(A \parr B) \times w} (A \parr B) \times (A \parr C) \xrightarrow{d} A \parr (B \times C).\end{array}$$

These arrows satisfy similar coherence conditions, and the following joint coherence conditions:

$$\begin{array}{ccc}(A \parr B) \times (C \parr D) & \xrightarrow{wd} & (A \times (C \parr D)) \parr B\\ \downarrow{\scriptstyle wd'} & & \downarrow{\scriptstyle wd' \parr B}\\ C \parr ((A \parr B) \times D) & \xrightarrow{C \parr wd} & C \parr (A \times D) \parr B\end{array}$$

and

$$\begin{array}{ccc}(A \parr B \parr C) \times D & \xrightarrow{wd''} & A \parr ((B \parr C) \times D)\\ \downarrow{\scriptstyle wd} & & \downarrow{\scriptstyle A \parr wd}\\ ((A \parr B) \times D) \parr C & \xrightarrow{wd'' \parr C} & A \parr (B \times D) \parr C.\end{array}$$

This follows again from naturality of *d* and coherence for *w* and *d*.

Proof of Lemma 3.1(1). Consider

$$\begin{array}{ccccc}(B^A \parr C \parr D) \times A & \xrightarrow{wd} & ((B^A \parr C) \times A) \parr D & \xrightarrow{wd \parr D} & (B^A \times A) \parr C \parr D\\ \downarrow{\scriptstyle (s \parr D) \times A} & & \downarrow{\scriptstyle (s \times A) \parr D} & & \downarrow{\scriptstyle \epsilon \parr C \parr D}\\ ((B \parr C)^A \parr D) \times A & \xrightarrow{wd} & ((B \parr C)^A \times A) \parr D & \xrightarrow{\epsilon \parr D} & B \parr C \parr D.\end{array}$$

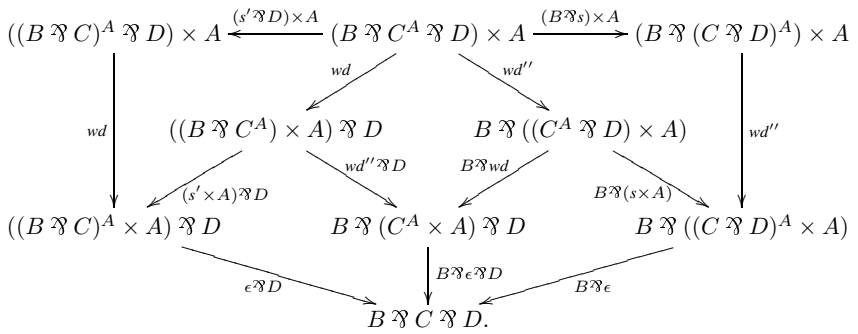
The left square commutes by naturality of *wd*. The right square commutes by definition of *s*. Currying along the top and right, one gets $s_{A,B,C \parr D}$. Currying along the left and bottom, one gets $s_{A,B \parr C,D} \circ (s_{A,B,C} \parr D)$. □

Proof of Lemma 3.1(2). Consider

$$\begin{array}{ccccc}B^A \times A & & \xrightarrow{\epsilon} & & B\\ & \swarrow^{l \times A} & \downarrow{\scriptstyle l} & & \downarrow{\scriptstyle l}\\ (B^A \parr \bot) \times A & \xrightarrow{wd} & (B^A \times A) \parr \bot & \xrightarrow{\epsilon \parr \bot} & B \parr \bot.\end{array}$$

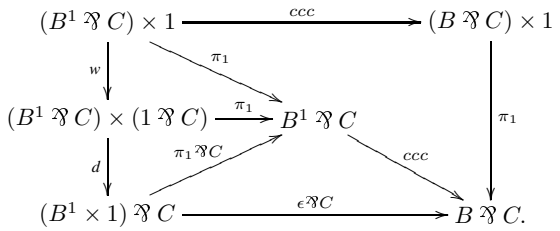
The triangle commutes by coherence of *wd*, the square by naturality of *l*. Currying clockwise, one gets $(l_B)^A$, currying counterclockwise, one gets $s_{A,B,\bot} \circ l_{B^A}$. □

Proof of Lemma 3.1(3). Consider



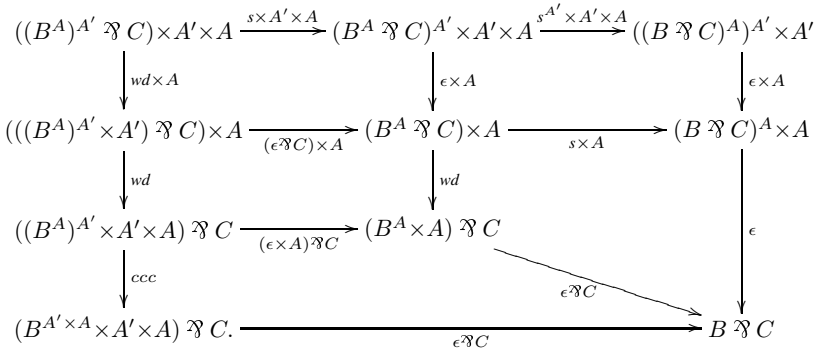
This commutes. Currying clockwise, one gets $s'_{A,B,C⅋D} \circ (B ⅋ s_{A,B,C})$. Currying counterclockwise, one gets $s_{A,B⅋C,D} \circ (s'_{A,B,C} ⅋ D)$. □

Proof of Lemma 3.1(4). Consider



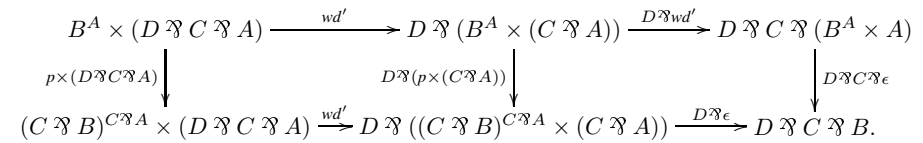
Clearly this commutes. Currying clockwise, one gets the natural ccc isomorphism, and currying counterclockwise, one gets $s_{1,B,C}$. □

Proof of Lemma 3.1(5). Consider



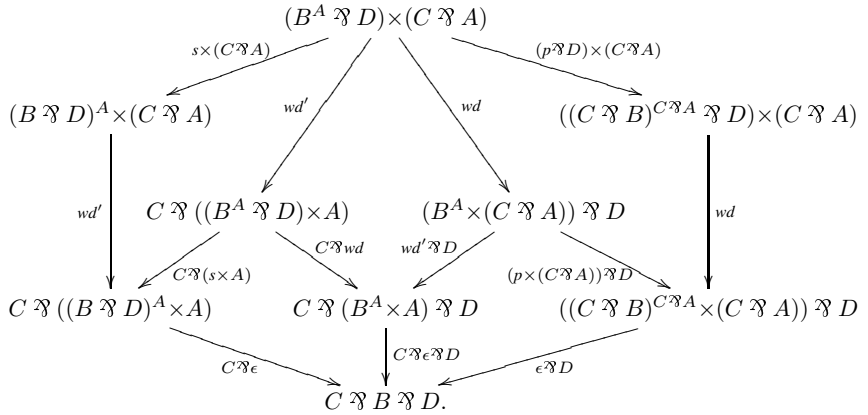
This commutes; currying counterclockwise, we get $s_{A'×A,B,C}$, whereas currying clockwise, we get $(s_{A,B,C})^{A'} \circ s_{A',B^A,C}$. □

Proof of Lemma 3.6(1). Consider



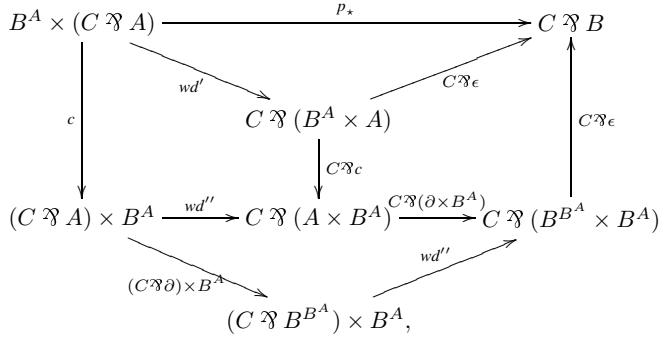
The left square commutes by naturality of wd'. The right square commutes by definition of p. Currying along the top and right, one gets $p_{A,B,D⅋C}$. Currying along the left and bottom, one gets $p_{C⅋A,C⅋B,D} \circ p_{A,B,C}$. □

Proof of Lemma 3.6(2). Consider



The three upper parts commute by naturality and coherence of wd and wd'. The lower left commutes by definition of s, and the lower right by definition of p. Currying along the left, one gets $p_{A,B⅋D,C} \circ s_{A,B,D}$. Currying along the right, one gets $s_{C⅋A,C⅋B,D} \circ (p_{A,B,C} ⅋ D)$. □

Proof of Lemma 3.6(3). First, for any A, B, and C, let $p^\star_\star : C ⅋ A \to (C ⅋ B)^{B^A}$ be the curry and uncurry of $p : B^A \to (C ⅋ B)^{C⅋A}$. Notice that



and thus, by currying and by definition of s',

$$p^\star_\star = C \parr A \xrightarrow{C \parr \partial} C \parr B^{B^A} \xrightarrow{s'} (C \parr B)^{B^A}.$$

To show the claim, it suffices to show that

$$\begin{array}{ccc} \bot^C \parr A & \xrightarrow{p^\star_\star} & (\bot^C \parr B)^{B^A} \\ \cong \uparrow & & \downarrow \cong \\ A^C & \xrightarrow{g \rhd \lambda f.\lambda c.f(gc)} & (B^C)^{B^A}. \end{array}$$

Now consider

$$\begin{array}{ccccccc} \bot^C \parr A & \xrightarrow{\bot^C \parr \partial} & \bot^C \parr B^{B^A} & & \xrightarrow{\quad s' \quad} & & (\bot^C \parr B)^{B^A} \\ {\scriptstyle s^{-1}}\uparrow & & \uparrow{\scriptstyle s^{-1}} & & & & \downarrow{\scriptstyle s} \\ (\bot \parr A)^C & \xrightarrow{(\bot \parr \partial)^C} & (\bot \parr B^{B^A})^C & \xrightarrow{s'} & ((\bot \parr B)^{B^A})^C & \xrightarrow{ccc} & ((\bot \parr B)^C)^{B^A} \\ \cong \uparrow & & \uparrow \cong & \nearrow_{\cong} & & & \downarrow \cong \\ A^C & \xrightarrow{\partial^C} & (B^{B^A})^C & & \xrightarrow{\quad ccc \quad} & & (B^C)^{B^A}. \end{array}$$

This commutes by naturality of s, by the first commutative diagram in Definition 2.11, and by Lemma 3.1(2). Along the top, we have $p^\star_\star$, and along the bottom, $g \rhd \lambda f.\lambda c.f(gc)$. □

Proof of Lemma 3.6(4). Consider

$$\begin{array}{ccccc} 1 \times (B \parr A) & & \xrightarrow{wd'} & & B \parr (1 \times A) \\ {\scriptstyle id^\star \times (B \parr A)}\downarrow & & & \searrow^{B \parr \pi_2} & {\scriptstyle B \parr (id^\star \times A)}\downarrow \\ A^A \times (B \parr A) & \xrightarrow{wd'} & B \parr (A^A \times A) & \xrightarrow{\epsilon} & B \parr A. \end{array}$$

The square commutes by naturality of wd'. The triangle commutes by cartesian-closed structure. Currying along the left and bottom, one gets $p_{A,A,B} \circ id^\star_A$. The arrow along the top is just π_2, so by currying one gets $id^\star_{B \parr A}$. □

Proof of Lemma 3.6 (p is natural in A, B, dinatural in central C). Clearly, the family of maps

$$B^A \times (C \parr A') \xrightarrow{w \times (C \parr A')} (C \parr B^A) \times (C \parr A') \xrightarrow{d} C \parr (B^A \times A')$$

is natural in A, A', B, and central C. Moreover,

$$C \parr (B^A \times A) \xrightarrow{C \parr \epsilon} C \parr B$$

is dinatural in A and natural in B and central C. Thus, it follows that $\tilde{\epsilon}_{A,B,C}$ is dinatural in A and natural in B and central C, and thus p is natural in A and B and dinatural in central C.

Lemma A.1. *$\epsilon : B^A \times A \to B$ is discardable.*

Proof. The axioms say that π_1 and π_2 are discardable, and that $i_{B^A} = (i_B)^{\diamond_A}$. From these two facts, it follows that $i_{B^A \times A} = (i_B)^{\diamond_A} \times i_A$. Now one has

$$\begin{array}{ccccc} \bot & & \xrightarrow{(i_B)^{\diamond_A} \times i_A} & & B^A \times A \\ {\scriptstyle \langle id, i_A \rangle}\downarrow & & & & \downarrow{\scriptstyle \epsilon} \\ \bot \times A & \xrightarrow{i_B \times \diamond_A} & B \times 1 & \xrightarrow{\pi_1} & B. \end{array}$$

Clearly, the counterclockwise arrow is i_B. □

Lemma A.2. *Let $ss_{A,B,C,D}$ and $ws_{A,B,C,D}$ be the maps*

$$\begin{aligned} ss_{A,B,C,D} &= A^C \parr B^D \xrightarrow{s} (A \parr B^D)^C \xrightarrow{s'^C} ((A \parr B)^D)^C, \\ ws_{A,B,C,D} &= (A \parr B) \times C \times D \xrightarrow{wd \times D} ((A \times C) \parr B) \times D \xrightarrow{wd''} (A \times C) \parr (B \times D). \end{aligned}$$

Then the (double) uncurry of ss is

$$\begin{aligned} ss_{\star\star} = (A^C \parr B^D) \times C \times D &\xrightarrow{ws_{A^C,B^D,C,D}} (A^C \times C) \parr (B^D \times D) \\ &\xrightarrow{\epsilon \parr (-)} A \parr (B^D \times D) \xrightarrow{A \parr \epsilon} A \parr B. \end{aligned}$$

Proof. Consider

$$\begin{array}{ccccc} (A^C \parr B^D) \times C \times D & \xrightarrow{s \times C \times D} & (A \parr B^D)^C \times C \times D & \xrightarrow{s'^C \times C \times D} & ((A \parr B)^D)^C \times C \times D \\ \downarrow{\scriptstyle wd \times D} & & \downarrow{\scriptstyle \epsilon \times D} & & \downarrow{\scriptstyle \epsilon \times D} \\ ((A^C \times C) \parr B^D) \times D & \xrightarrow{(\epsilon \parr B^D) \times D} & (A \parr B^D) \times D & \xrightarrow{s' \times D} & (A \parr B)^D \times D \\ \downarrow{\scriptstyle wd''} & & \downarrow{\scriptstyle wd''} & & \downarrow{\scriptstyle \epsilon} \\ (A^C \times C) \parr (B^D \times D) & \xrightarrow{\epsilon \parr (-)} & A \parr (B^D \times D) & \xrightarrow{A \parr \epsilon} & A \parr B. \end{array}$$

The top left square commutes by definition of s. The top right square commutes by naturality of ϵ. The bottom left square commutes by naturality of $wd''_{A,B,C}$ in discardable A, and because ϵ is discardable by Lemma A.1. The bottom right square commutes by definition of s'. □

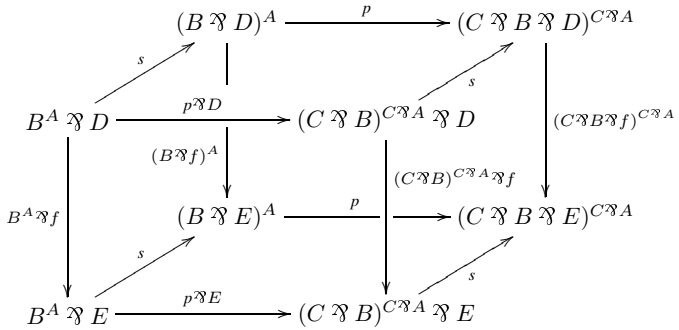
Lemma A.3. *The following two diagrams commute:*

$$\begin{array}{ccccc} (B \parr B) \times (C \parr A) & & \xrightarrow{- \times \Delta} & & (B \parr B) \times (C \parr A) \times (C \parr A) \\ \downarrow{\scriptstyle wd'} & & & & \downarrow{\scriptstyle - \times (wd''; C \parr wd')} \\ C \parr ((B \parr B) \times A) & & & & (B \parr B) \times (C \parr C \parr (A \times A)) \\ \downarrow{\scriptstyle C \parr ((B \parr B) \times \Delta)} & & & & \downarrow{\scriptstyle wd'} \\ C \parr ((B \parr B) \times A \times A) & & \xleftarrow{\nabla \parr -} & & C \parr C \parr ((B \parr B) \times A \times A) \end{array}$$

$$
\begin{array}{ccc}
(B \parr B) \times (C \parr A) \times (C \parr A) & \xrightarrow{ws} & (B \times (C \parr A)) \parr (B \times (C \parr A)) \\
\downarrow{\scriptstyle -\times(wd'';C\parr wd')} & & \downarrow{\scriptstyle wd'\parr wd'} \\
(B \parr B) \times (C \parr C \parr (A \times A)) & & C \parr (B \times A) \parr C \parr (B \times A) \\
\downarrow{\scriptstyle wd'} & & \downarrow{\scriptstyle -\parr c\parr -} \\
C \parr C \parr ((B \parr B) \times A \times A) & \xrightarrow{-\parr ws} & C \parr C \parr (B \times A) \parr (B \times A).
\end{array}
$$

Proof. Two straightforward diagram chases from the definitions. □

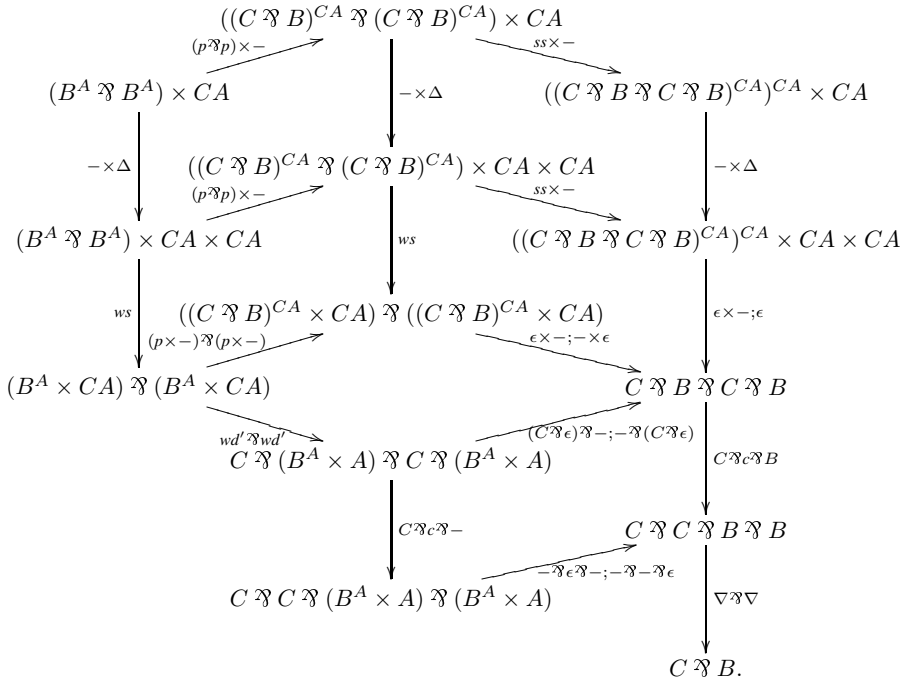
Proof of Lemma 3.6 (p is central). Let $f : D \to E$. Consider the following cube:



The top and bottom faces commute by Lemma 3.6(2). The left and right faces commute by naturality of s. The back face commutes by naturality of p. Thus, the front commutes, which shows that p is central. □

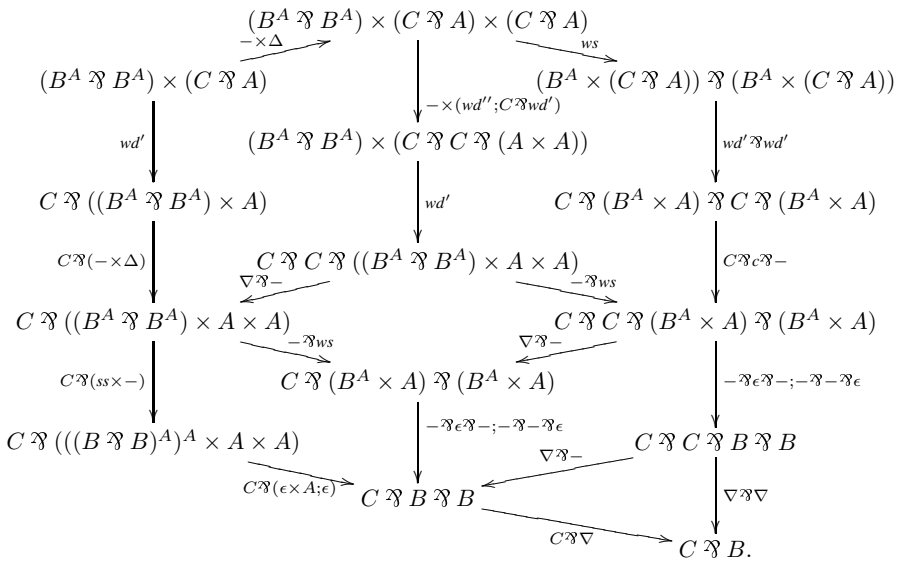
Proof of Lemma 3.6 (p is copyable). Consider the following diagram, where CA abbreviates

$(C \parr A)$.

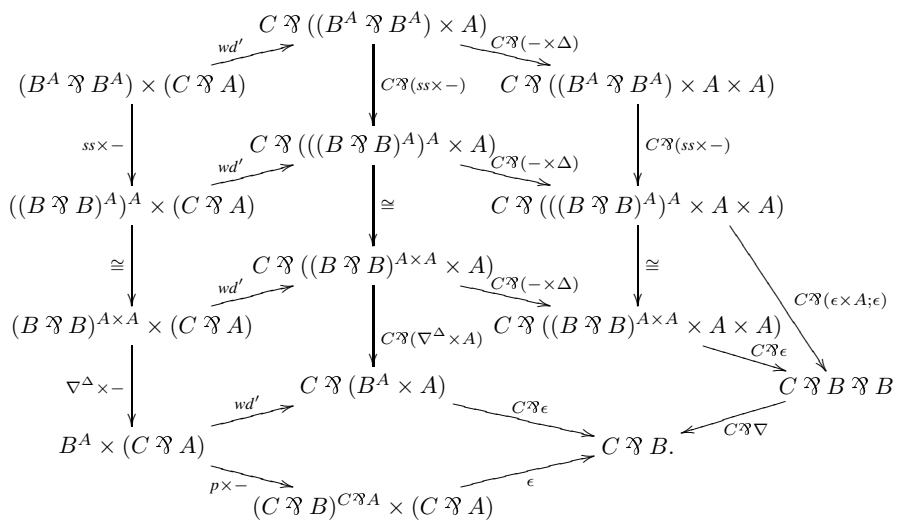


The two top squares commute trivially. The next square commutes by naturality of ws and the fact that p is central. The next square commutes by Lemma A.2. The big triangle commutes by definition and centrality of p. The parallelogram commutes by naturality of c. Clockwise, one has the curry of $(p \parr p); \nabla$, which can be seen with a few simple ccc manipulations. Along the

counterclockwise arrow, we continue with the diagram



The two top cells commute by Lemma A.3. The rest commutes by the fact that ∇ is central, and by Lemma A.2. Along the counterclockwise arrow, we continue with the diagram



The three left squares commute by naturality of wd'. The bottom triangle commutes by definition of p; everything else commutes by ccc operations. Finally, currying counterclockwise, we get

$$B^A \parr B^A \xrightarrow{\nabla} B^A \xrightarrow{p} (C \parr B)^{C\parr A}.$$

Thus, the last three diagrams show that $(p \parr p);\nabla = \nabla;p$, and thus that p is copyable. □

Acknowledgments

Most of this research was done in the spring of 1998 while I was visiting BRICS, Basic Research in Computer Science, the Centre of the Danish National Research Foundation. I thank my hosts at BRICS for a productive semester. I am indebted to John Power for his encouragement in an early phase of this work. I thank Olivier Danvy for pointing me to Filinski's work and for offering helpful advice. Thanks to Hayo Thielecke for his explanations on the nature of continuations, and to Carsten Führmann, Alan Jeffrey, and Paul Levy, who have contributed to this work through stimulating discussions.

References

B. Agapiev and E. Moggi. Declarative continuations and monads. Unpublished draft, 1991.

P. De Groote. A CPS-translation of the $\lambda\mu$-calculus. In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming – CAAP '94*, Springer LNCS 787, pages 85–99, Edinburgh, 1994a.

P. De Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. Springer LNCS 822, 1994b.

M. Felleisen. *The calculi of λ_v-conversion: A syntactic theory of control and state in imperative higher order programming languages*. PhD thesis, Indiana University, 1986.

M. Felleisen and R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103:235–271, 1992.

A. Filinski. Declarative continuations and categorical duality. Master's thesis, DIKU, Computer Science Department, University of Copenhagen, Aug. 1989. DIKU Report 89/11.

C. Führmann. Direct-style and continuation-passing style models of control. Available from http://www.dcs.ed.ac.uk/home/car/research.htm, November 1998.

C. Führmann. Direct models of the computational lambda-calculus. In *Proceedings MFPS XV*, Electronic Notes in Theoretical Computer Science 20, 1999.

T. G. Griffin. A formulae-as-types notion of control. In *POPL '90: Proceedings of the 17th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1990.

M. Hofmann and T. Streicher. Continuation models are universal for $\lambda\mu$-calculus. In *Proceedings of the Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 387–397, 1997.

A. Jeffrey. Premonoidal categories and a graphical view of programs. Preprint, Dec. 1997.

G. M. Kelly. On Mac Lane's conditions for coherence of natural associativities, commutativities, etc. *J. Algebra*, 1:397–402, 1964.

Y. Lafont, B. Reus, and T. Streicher. Continuations semantics or expressing implication by negation. Technical Report 93-21, University of Munich, 1993.

J. Lambek and P. J. Scott. *An Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics 7. Cambridge University Press, New York, 1986.

S. Mac Lane. Natural associativity and commutativity. *Rice University Studies*, 49:28–46, 1963.

E. Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, Edinburgh University, Department of Computer Science, 1988.

C.-H. L. Ong. A semantic view of classical proofs: Type-theoretic, categorical, and denotational characterizations. In *Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science*, pages 230–241, 1996.

C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proceedings of the Symposium on Principles of Programming Languages*, pages 215–227, 1997.

M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning, St. Petersburg*, Springer LNCS 624, pages 190–201, 1992.

G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1:125–159, 1975.

J. Power and E. Robinson. Premonoidal categories and notions of computation. *Math. Struct. in Computer Science*, 7(5):445–452, 1997.

D. Pym and E. Ritter. On the semantics of classical disjunction. Preprint, 1998.

P. Selinger. An implementation of the call-by-name $\lambda\mu\nu$-calculus. Preprint. Available from the Hypatia Electronic Library, 1998.

T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, Nov. 1998.

H. Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997.