Call-by-Value is Dual to Call-by-Name Reloaded

Philip Wadler
Avaya Labs
wadler@avaya.com

ABSTRACT

We consider the relation of the dual calculus of Wadler (2003) to the $\lambda\mu$-calculus of Parigot (1992). We give a translation from the $\lambda\mu$-calculus into the dual calculus, and an inverse translation from the dual calculus back into the $\lambda\mu$-calculus. The translations form an equational correspondence as defined by Sabry and Felleisen (1993). Translating from $\lambda\mu$ to dual and then 'reloading' from dual back into $\lambda\mu$ yields a term equal to the original term. Composing the translations with duality on the dual calculus yields an involutive notion of duality on the $\lambda\mu$-calculus. A previous notion of duality on the $\lambda\mu$-calculus has been suggested by Selinger (2001), but it is not involutive.

Note. This paper uses color to clarify the relation of types and terms, and of source and target calculi. If the URL below is not in blue please download the color version from

http://www.research.avayalabs.com/user/wadler

or google 'wadler dual reloaded'.

Categories and Subject Descriptors

F.4.1 [**Theory of Computation**]: Mathematical Logic

General Terms

Languages, Theory

Keywords

Curry-Howard correspondence, sequent calculus, natural deduction, De Morgan dual, logic, lambda calculus, lambda mu calculus

1. INTRODUCTION

Sometimes less is more. Implication is a key connective of logic, but for some purposes it is better to define it in terms of other connectives, taking $A \supset B \equiv \neg A \vee B$ or $A \supset B \equiv \neg(A \,\&\, \neg B)$. This is helpful if one wishes to understand de Morgan duality. The dual of & is $\vee$, and $\neg$ is self dual, but the dual of implication is the difference operator, $B - A \equiv B \,\&\, \neg A$ or $B - A \equiv \neg(\neg B \vee A)$, which is not particularly familiar.

Church (1932) introduced the call-by-name λ-calculus, and a few years later Bernays (1936) proposed the call-by-value variant. A line of work, including that of Filinski (1989), Griffin (1990), Parigot (1992), Danos, Joinet, and Schellinx (1995), Barbanera and Berardi (1996), Streicher and Reuss (1998), Selinger (1998,2001), and Curien and Herbelin (2000), has led to a startling conclusion: call-by-value is the de Morgan dual of call-by-name.

Wadler (2003) presents a dual calculus that corresponds to the classical sequent calculus of Gentzen (1935) in the same way that the lambda calculus of Church (1932,1940) corresponds to the intuitionistic natural deduction of Gentzen (1935). The calculus possesses an involutive duality, which takes call-by-value into call-by-name and vice-versa. A key to achieving this is to not take implication as primitive, but to define it by taking $A \supset B \equiv \neg A \vee B$ under call-by-name, or $A \supset B \equiv \neg(A \,\&\, \neg B)$ under call-by-value.

Wadler (2003) included a discussion of call-by-value and call-by-name CPS translations from the dual calculus into the λ-calculus. Here we complete the story by discussing a translation from the $\lambda\mu$-calculus of Parigot (1992) into the dual calculus, together with an inverse translation. We will show that there is a translation from the $\lambda\mu$-calculus into the dual calculus which forms an *equational correspondence*, as defined by Sabry and Felleisen (1993).

Say we have a source and target calculus with equations defined on them, writing

$$M =_v N$$

for equality in the source, and

$$M =^v N$$

for equality in the target. Further, say we have translations between them, such that

$$(M)^* \equiv N$$

converts source term M to a target term N, and

$$(N)_* \equiv M$$

converts target term N to source term M. We have an *equational correspondence* if the following four conditions hold.

- The translation from source to target preserves equations,
$$M =_v N \text{ implies } (M)^* =^v (N)^*,$$
with M, N source terms.
- The translation from target to source preserves equations,
$$M =^v N \text{ implies } (M)_* =_v (N)_*,$$
with M, N target terms.
- Translating for source to target and then 'reloading' from target to source yields a term equal to the original term,
$$((M)^*)_* =_v M,$$
with M a source term.
- Translating for target to source and then 'reloading' from source to target yields a term equal to the original term,
$$((M)_*)^* =^v M,$$
with M a target term.

The existence of an equational correspondence shows in a strong sense that the translation is both *sound* and *complete* with respect to equations. In particular an equation holds in the source if and only if its translation holds in the target.

Wadler (2003) shows that there is a CPS translation from dual calculus into λ-calculus that forms a *reflection*, as discussed by Sabry and Wadler (1997). A reflection is a special case of both a *Galois connection*, as discussed by Sabry and Wadler (1997), and of a *Lagois connection*, as discussed by Melton, Schröder, and Strecker (1994). Every Galois or Lagois connection is an equational correspondence (where, as usual, equality is the reflexive, symmetric, and transitive closure of reduction).

It is easy to see that the composition of two equational correspondences is also an equational correspondence. Reflections and Galois connections also compose, and Lagois connections compose under some circumstances.

The translation from $\lambda\mu$-calculus into dual calculus is an equational correspondence for both call-by-value and call-by-name, and the CPS translation for dual calculus comes in dual call-by-value and call-by-name variants, both of which are reflections, and hence equational correspondences. Their composition yields the usual call-by-value and call-by-name CPS translations for $\lambda\mu$, as studied by Hoffman and Streicher (1997) and Selinger (2001); and it follows immediately that both of these are equational correspondences.

Fujita (2003) also shows that the call-by-value CPS translation for $\lambda\mu$-calculus is an equational correspondence; but says nothing about call-by-name. The advantage of the proof here is that the CPS translation for $\lambda\mu$ can be computed by composing other translations, and that its properties follow immediately from its construction by composition rather than requiring separate proof.

Duality is a translation that takes the dual calculus into itself. Since it is an involution (the dual of the dual is the identity), duality is trivially an equational correspondence. We may derive a duality transform from $\lambda\mu$-calculus to itself by forming the threefold composition of (i) the translation from $\lambda\mu$-calculus to dual calculus with (ii) the duality translation from dual calculus to itself with (iii) the reloading translation from dual calculus back to $\lambda\mu$-calculus; and follows immediately that this is an equational correspondence. The same duality transform works for both call-by-value and call-by-name.

Selinger (2001) also presents a duality transformation for $\lambda\mu$-calculus, and shows that it is an involution up to equality in a control category. Selinger's duality required some cleverness to construct — it answered an open question of Streicher and Reuss (1998). The advantage of the proof here is that duality for $\lambda\mu$ can be computed by composing other translations, and that its properties follow immediately from its construction by composition rather than requiring separate proof. Also, the work here uses purely syntactic techniques, depending only on equations in the $\lambda\mu$ and dual calculi, with no reference to control categories or other semantic frameworks.

This paper contains almost entirely new material as compared with Wadler (2003). The description of the dual calculus overlaps with that paper, but the relationship with $\lambda\mu$ is entirely new, as is the treatment of η laws.

2. THE $\lambda\mu$-CALCULUS

The syntax and type rules of the $\lambda\mu$-calculus are shown in Figure 1. Following Parigot (1993), we distinguish two main constructs, *terms* and *statements* (Parigot called these *unnamed terms* and *named terms*.)

As usual, we require the body of a μ-abstraction to be a statement. However, we also take the unusual step of requiring the body of a λ-abstraction to be a statement. General functions take an argument and return a result, and correspond to an implication $A \supset B$. Our restricted functions take an argument but return no result, and correspond to a negation $\neg A \equiv A \supset \bot$. The restriction is not as severe as it might at first seem, since general functions may be defined it terms of our restricted functions, just as one can define implication in terms of negation by taking $A \supset B \equiv \neg A \vee B$ or $A \supset B \equiv \neg A \vee B \neg(A \,\&\, \neg B)$.

Let A, B range over types. A type is atomic X; a conjunction $A \& B$; a disjunction $A \vee B$; or a negation $\neg A$. Let x, y, z range over variables, and α, β, γ range over covariables.

Let M, N, O range over terms, and S, T range over statements. A term is a variable x; a λ-abstraction $\lambda x.\, S$; or a μ-abstration $\mu\alpha.\, S$. A statement is a function application $O\, M$ or a covariable application $[\alpha]\, M$. The computational interpretation of a μ-abstraction $\mu\alpha.\, S$ is to bind the covariable α and then evaluate statment S; if during evaluation of S the covariable α is applied to a value, then that value is returned as the value of the μ-abstraction; this is similar to the behaviour of *callcc* in Scheme.

We also have products and sums. Products are constructed with pairing $\langle M, N \rangle$ and decontstructed with projections $\mathrm{fst}\, O$ and $\mathrm{snd}\, O$. Following Selinger (2001), we construct sums with a variant of the mu abstraction $\mu[\alpha, \beta].\, S$, and deconstruct sums with a variant of covariable application $[\alpha, \beta]\, O$. The term $\mu[\alpha, \beta].\, S$ constructs a sum: if α is passed a value of type A then the μ-abstraction returns a left injection into the sum type $A \vee B$, and if β is passed a value of type B then the μ-abstraction returns a right injection

into the sum type $A \vee B$. Conversely, the statement $[\alpha, \beta]\, O$ deconstructs a sum; the term O has a sum type $A \vee B$, and if it returns a left summand then covariable α is passed the value of type A, while if it returns a right summand then covariable β is passed the value of type B.

Substitution of a term for a variable is standard, but substitution for a covariable is slightly tricky. The notation used here is adapted from Selinger (2001).

DEFINITION 2.1. *(Substitution for a covariable) Let S be a statement, α a covariable of type A, and $T\{-\}$ be a statement context with a hole accepting a term of type A. We write*

$$S\{T\{-\}/[\alpha]\,\{-\}\}$$

for the substitution that makes the recursive replacements

$$\begin{array}{lcl} [\alpha]\, M & \mapsto & T\{M\}, \\ [\alpha, \beta]\, O & \mapsto & T\{\mu\alpha.\, [\alpha, \beta]\, O\}, \\ [\beta, \alpha]\, O & \mapsto & T\{\mu\alpha.\, [\beta, \alpha]\, O\}. \end{array}$$

Call-by-value reductions are shown in Figure 3 and call-by-name reductions are shown in Figure 4. We write $\longrightarrow_v$ and $\longrightarrow_n$ for the reflexive and transitive closure of the reductions shown in the figures, and $=_v$ and $=_n$ for the reflexive, symmetric, and transitive closure.

Let V, W range over values. A value is a variable, a pair of values, an injection of a value, a function, or a projection from a value.

The rules are grouped as β rules, which reduce a deconstructor applied to a contructor; η rules, which introduce a constructor applied to a deconstructor (this may enable β reductions elsewhere); ν rules, which introduce names for terms; and ς rules, which perform commuting conversions.

The (ν) rules are similar to the reductions (let.1) and (let.2) in the λ_c-calculus of Moggi (1988).

$$\begin{array}{lcl} (\text{let.1}) & M\, N & \longrightarrow_c \quad \text{let } x = M \text{ in } x\, N \\ (\text{let.2}) & V\, N & \longrightarrow_c \quad \text{let } y = N \text{ in } V\, y \end{array}$$

These reductions correspond to the operation of introducing names for subterms in continuation passing style, as explained by Sabry and Wadler (1997).

The reductions were derived by putting an order on the equations given by Selinger (2001); some of Selinger's equations could be simplified because of the restriction on functions in our formulation. The η expansions should be applied only in a context where the introduced constructor is not immediately deconstructed; this avoids the regress of applying the same expansion infinitely. Restrictions to achieve this in the call-by-name case are discussed by Pym and Ritter (2001), who prove confluence and strong normalization for a system similar to the one presented here.

The restriction on functions does not lose expressive power, since general functions can be defined in terms of restricted functions. However, different definitions must be used for call-by-value or call-by-name.

PROPOSITION 2.2. *Under call-by-value, functions may be defined by*

$$\begin{array}{lcl} A \supset B & \equiv & \neg(A \,\&\, \neg B) \\ \lambda x.\, N & \equiv & \lambda z.\, (\lambda x.\, \mathit{snd}\, z\, N)\, (\mathit{fst}[z]) \\ O\, M & \equiv & \mu\beta.\, O\, \langle M, \lambda y.\, [\beta]\, y\rangle \end{array}$$

validating the reduction rules

$$\begin{array}{lcll} (\beta\supset) & (\lambda x.\, N)\, V & \longrightarrow_v & N\{V/A\} \\ (\eta\supset) & V : A \supset B & \longrightarrow_v & \lambda x.\, V\, x, \end{array}$$

and where the translation of a function abstraction is a value.

PROPOSITION 2.3. *Under call-by-name, functions may be defined by*

$$\begin{array}{lcl} A \supset B & \equiv & \neg A \vee B \\ \lambda x.\, N & \equiv & \mu[\gamma, \beta].\, [\gamma]\, \lambda x.\, [\beta]\, N \\ O\, M & \equiv & \mu\beta.\, (\mu\gamma.\, [\gamma, \beta]\, O)\, M \end{array}$$

validating the reduction rules

$$\begin{array}{lcll} (\beta\supset) & (\lambda x.\, N)\, M & \longrightarrow_n & N\{M/A\} \\ (\eta\supset) & M : A \supset B & \longrightarrow_n & \lambda x.\, M\, x. \end{array}$$

3. THE DUAL CALCULUS

Figure 2 presents the syntax and inference rules of the dual calculus. Types, variables, and covariables are the same as the $\lambda\mu$-calculus.

Let M, N range over terms, which yield values. A term is either a variable x; a pair $\langle M, N\rangle$; an injection on the left or right of a sum $\langle M\rangle\mathrm{inl}$ or $\langle N\rangle\mathrm{inr}$; a complement of a coterm $[K]\mathrm{not}$; a function abstraction $\lambda x.\, N$, with x bound in N; or a covariable abstraction $(S).\alpha$, with α bound in S.

Let K, L range over coterms, which consume values. A coterm is either a covariable α; a projection from the left or right of a product $\mathrm{fst}[K]$ or $\mathrm{snd}[L]$; a case $[K, L]$; a complement of a term $\mathrm{not}\langle M\rangle$; a function application $M \mathbin{@} L$; or a variable abstraction $x.(S)$, with x bound in S.

Finally, let S, T range over statements. A statement is a cut of a term against a coterm, $M \bullet K$. Note that angle brackets always surround terms, square brackets always surround coterms, and round brackets always surround statements. Curly brackets are used for substitution and holes in contexts.

The type rules given here differ slightly from Wadler (2003), in that they are presented in syntax-directed form; so thinning, exchange, and contraction are built into the form of the rules rather than given as separate structural rules.

A cut of a term against a variable abstraction, or a cut of a covariable abstraction against a coterm, corresponds to substitution. This suggests the following reduction rules.

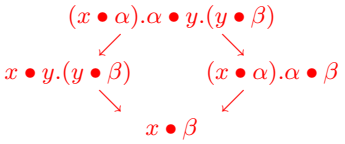
$$\begin{array}{lcl} (\beta\mathrm{L}) & M \bullet x.(S) & \longrightarrow \quad S\{M/x\} \\ (\beta\mathrm{R}) & (S).\alpha \bullet K & \longrightarrow \quad S\{K/\alpha\} \end{array}$$

Here substitution in a statement of a term for a variable is written $S\{M/x\}$, and substitution in a statement of a coterm for a covariable is written $S\{K/\alpha\}$.

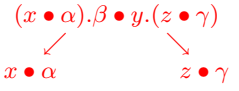
A critical pair occurs when a covariable abstraction is cut against a variable abstraction.

$$(S).\alpha \bullet x.(T)$$

Sometimes such reductions are confluent.



But sometimes they are not.



To restore confluence we must limit reductions, and this is achieved by adopting call-by-value or call-by-name.

Call-by-value only reduces a cut of a value against a variable abstraction, but reduces a cut of a covariable abstraction against any coterm.

$$\begin{array}{lll} (\beta\mathrm{L}) & V \bullet x.(S) & \longrightarrow^v S\{V/x\} \\ (\beta\mathrm{R}) & (S).\alpha \bullet K & \longrightarrow^v S\{K/\alpha\} \end{array}$$

Value V replaces term M in rule (βL). A value cannot be a covariable abstraction, so this avoids the critical pair.

Call-by-name only reduces a cut of a covariable abstraction against a covalue, but reduces a cut of any coterm against a variable abstraction.

$$\begin{array}{lll} (\beta\mathrm{L}) & V \bullet x.(S) & \longrightarrow^v S\{V/x\} \\ (\beta\mathrm{R}) & (S).\alpha \bullet K & \longrightarrow^v S\{K/\alpha\} \end{array}$$

Covalue P replaces coterm K in rule (βR). A covalue cannot be a variable abstraction, so this avoids the critical pair.

In λ-calculus, the move from call-by-value to call-by-name generalizes values to terms. In dual calculus, the move from call-by-value to call-by-name generalizes values to terms but restricts coterms to covalues, clarifying the duality.

Call-by-value reductions are shown in Figure 5 and call-by-name reductions are shown in Figure 6. We write $\longrightarrow^v$ and $\longrightarrow^n$ for the reflexive and transitive closure of the reductions shown in the figures, and $=^v$ and $=^n$ for the reflexive, symmetric, and transitive closure.

Let V, W range over values. A value is a variable, a pair of values, a left or right injection of a value, or any complement. The fact that any complement is a value is analogous to the fact that any function is a value in the λ_v calculus. Unlike the call-by-value $\lambda\mu$ calculus, we do not define the projection of a value to be a value, but the same effect is achieved by the ($\varsigma\vee$) rules.

Let P, Q range over covalues. A covalue is a covariable, a first or second projection of a covalue, a case over a pair of covalues, or any complement. Covalues correspond to a strict context, one that is guaranteed to demand the value passed to it.

As before, the reduction rules are grouped into β, η, ν, and ς rules. Note that the ν rules are called ς rules in Wadler (2003).

4. TRANSLATIONS

We now consider the translation from the $\lambda\mu$-calculus to the dual calculus and its inverse translation.

Definition 4.1. *The translation from the $\lambda\mu$-calculus into the dual calculus is given in Figure 7. It consists of three operations,*

$$(M)^*, \quad (M)^*\{P\}, \quad (S)^*.$$

- *If M is a $\lambda\mu$ term of type A, then $(M)^*$ is a dual term of type A.*
- *If M is a $\lambda\mu$ term of type A and P is a dual cotermm of type A, then $(M)^*\{P\}$ is a dual statement.*
- *If S is a $\lambda\mu$ statement, then $(S)^*$ is a dual statement.*

Definition 4.2. *The translation from the dual calculus into the $\lambda\mu$-calculus is given in Figure 8. It consists of three operations,*

$$(M)_*, \quad (K)_*\{O\}, \quad (S)_*.$$

- *If M is a dual term of type A, then $(M)_*$ is a $\lambda\mu$ term of type A.*
- *If K is a dual coterm of type A, and O is a $\lambda\mu$ term of type A, then $(K)_*\{O\}$ is a dual statement.*
- *If S is a dual statement, then $(S)_*$ is a $\lambda\mu$ statement.*

Although the translations preserve β reductions, they do not preserve all η, ν, and ς reductions. However, they do preserve equalities, where equality is defined as the symmetric and transitive closure of reduction.

In general, these translatios do not preserve reductions, but they do preserve equalities. We now present the detailed results to show that the translations form an equational correspondence between the call-by-value $\lambda\mu$ calculus and the call-by-value dual calculus.

Proposition 4.3. *($\lambda\mu$ reloaded) Translating from the $\lambda\mu$-calculus into the dual calculus and then 'reloading' into the $\lambda\mu$-calculus gives a term equal to the original under call-by-value,*

$$\begin{array}{lll} ((M)^*)_* & =_v & M \\ ((M)^*\{K\})_* & =_v & (K)_*\{M\} \\ ((S)^*)_* & =_v & S, \end{array}$$

with M a term in $\lambda\mu$, K a coterm in , and S a statement in $\lambda\mu$.

The first line follows immediately from the second, since

$$(M)^* \equiv ((M)^*\{\alpha\}).\alpha = ((\alpha)_*\{M\}).\alpha = (M \bullet \alpha).\alpha = M.$$

The second and third lines are shown by case analysis on terms and statements of $\lambda\mu$.

Proposition 4.4. *(dual reloaded) Translating from the dual calculus into the $\lambda\mu$-calculus and then 'reloading' into the dual calculus gives a term equal to the original under call-by-value,*

$$\begin{array}{lll} ((M)_*)^* & =^v & M \\ ((K)_*\{O\})^* & =^v & (O)^* \bullet K \\ ((S)_*)^* & =^v & S, \end{array}$$

with M a term in dual, K a coterm in dual, O a term in $\lambda\mu$, and S a statement in dual.

The three lines are shown by case analysis on terms, coterms, and statements of dual.

Proposition 4.5. *($\lambda\mu$ to dual preserves equalities) Translating from the $\lambda\mu$-calculus into the dual calculus preserves call-by-value equalities,*

$$\begin{array}{lll} M =_v N & \text{implies} & (M)^* =^v (N)^* \\ M =_v N & \text{implies} & (M)^*\{K\} =^v (N)^*\{K\} \\ S =_v T & \text{implies} & (S)^* =^v (T)^*, \end{array}$$

with M, N terms in $\lambda\mu$, K a coterm in dual, and S, T statements in $\lambda\mu$.

The first line follows immediately from the second and Proposition 4.3. The second and third lines are shown by case analysis on the reductions of $\lambda\mu$ that apply to terms and statements respectively.

PROPOSITION 4.6. *(dual to $\lambda\mu$ preserves equalities) Translating from the dual calculus into the $\lambda\mu$-calculus preserves call-by-value equalities,*

$$\begin{array}{lll} M =^v N & \textit{implies} & (M)_* =_v (N)_* \\ K =^v L & \textit{implies} & (K)_*\{O\} =_v (L)_*\{O\} \\ S =^v T & \textit{implies} & (S)_* =_v (T)_*, \end{array}$$

with M, N terms in dual, K, L coterms in dual, O a term in $\lambda\mu$, and S, T statement in dual.

The three lines are shown by case analysis on the reductions of dual that apply to terms, coterms, and statements respectively.

The four propositions above also hold for call-by-name. The restatement is easy, simply replace $=_v$ and $=^v$ everywhere by $=_n$ and $=^n$. However, while the structure of the proofs is essentially the same, the new sets of reductions require that one repeat the proofs entirely, since there is no simple, systematic relation between the call-by-value and call-by-name reductions of $\lambda\mu$.

However, there is a systematic relation between the call-by-value and call-by-name reductions of dual. We next consider how to characterize and exploit this regularity.

5. DUALITY

We now review the results about duality for the dual from Wadler (2003), and use these to derive similar results concering duality for the $\lambda\mu$-calculus.

The dual calculus is designed to exploit duality. Variables are dual to covariables, pairs are duals to sums, complement is self dual, term abstraction is dual to coterm abstraction, and cut is self dual.

This can be captured in a translation from the dual calculus into itself. The translation is involutive – that is, it is its inverse – and it carries call-by-value reductions into call-by-name, and vice versa. So it is an equational correspondence.

We assume a one-to-one correspondence between variables and covariables. Each variable x corresponds to a covariable $\bar{x}$, and each covariable α corresponds to a variable $\bar{\alpha}$, such that $\bar{\bar{x}} \equiv x$ and $\bar{\bar{\alpha}} \equiv \alpha$. For instance, we might take $\bar{x} \equiv \alpha, \bar{y} \equiv \beta, \bar{z} \equiv \gamma$, and hence $\bar{\alpha} = x, \bar{\beta} = y, \bar{\gamma} = z$.

DEFINITION 5.1. *The duality translation from the dual calculus to itself is given in Figure 9. It consists of three operations,*

$$(M)^\circ, \quad (K)^\circ, \quad (S)^\circ.$$

- *If M is a dual term of type A, then $(M)^\circ$ is a dual coterm of type A.*
- *If K is a dual coterm of type A, and $(K)^\circ$ is a dual term of type A.*
- *If S is a dual statement, then $(S)^\circ$ is a dual statement.*

It is immediate from the definition that duality is its own inverse.

PROPOSITION 5.2. *(Involution) Duality is an involution up to identity,*

$$\begin{array}{lll} ((A)^\circ)^\circ & \equiv & A \\ ((M)^\circ)^\circ & \equiv & M \\ ((K)^\circ)^\circ & \equiv & K \\ ((S)^\circ)^\circ & \equiv & S, \end{array}$$

with A a type of dual, M a term of dual, K a coterm of dual, and S a statement of dual.

It is easy to confirm that the type rules come in dual pairs, &R is dual to ∨L, ∨R is dual to &L, ¬R is dual to ¬L, the Id rules are dual, and Cut is dual to itself.

PROPOSITION 5.3. *Duality preserves types,*

$$\begin{array}{lll} \Gamma \rightarrow \Theta \mid M : A & \textit{iff} & (M)^\circ : (A)^\circ \mid (\Theta)^\circ \rightarrow (\Gamma)^\circ \\ K : A \mid \Gamma \rightarrow \Theta & \textit{iff} & (\Theta)^\circ \rightarrow (\Gamma)^\circ \mid (K)^\circ : (A)^\circ \\ \Gamma \mid S \mapsto \Theta & \textit{iff} & (\Theta)^\circ \mid (S)^\circ \mapsto (\Gamma)^\circ, \end{array}$$

with A a type of dual, M a term of dual, K a coterm of dual, and S a statement of dual.

For the dual calculus, call-by-value is dual to call-by-name. This is easily confirmed by inspection of the reduction rules; indeed, it was the principle guiding their design.

PROPOSITION 5.4. *(Call-by-value is dual to call-by-name) Duality takes call-by-value reductions on dual, into call-by-name reductions, and vice versa.*

$$\begin{array}{lll} M \longrightarrow^v N & \textit{iff} & (M)^\circ \longrightarrow^n (N)^\circ \\ K \longrightarrow^v L & \textit{iff} & (K)^\circ \longrightarrow^n (L)^\circ \\ S \longrightarrow^v T & \textit{iff} & (S)^\circ \longrightarrow^n (T)^\circ, \end{array}$$

with M, N terms of dual, K, L coterms of dual, and S, T statements of dual.

An immediate consequence of the above is that duality is an equational correspondence between the call-by-value dual calculus and the call-by-name dual calculus.

We now extend the above results from the dual calculus to the $\lambda\mu$-calculus.

Using the translations of the previous section, we can compute duals for the $\lambda\mu$-calculus by translating from $\lambda\mu$ to dual, taking the dual, and then 'reloading' back into $\lambda\mu$.

DEFINITION 5.5. *The duality transformation from the $\lambda\mu$ calculus to itself is given in Figure 10. It consists of two operations, defined as follows,*

$$\begin{array}{lll} (M)_\circ\{O\} & \equiv & (((M)^*)^\circ)_*\{O\} \\ (S)_\circ & \equiv & (((S)^*)^\circ)_* \end{array}$$

- *If M is a $\lambda\mu$ term of type A and O is a $\lambda\mu$ term of type A, then $(M)_\circ\{O\}$ is a $\lambda\mu$ statement.*
- *If S is a $\lambda\mu$ statement, then $(S)_\circ$ is a $\lambda\mu$ statement.*

In effect, we compose three equational correspondences (from $\lambda\mu$ to dual, from dual to itself, and from dual to $\lambda\mu$) to yield a new equational correspondence (from $\lambda\mu$ to itself).

It follows immediately that duality on $\lambda\mu$ takes call-by-value into call-by-name.

PROPOSITION 5.6. *(Call-by-value is dual to call-by-name, reloaded) Duality takes call-by-value equalities on $\lambda\mu$ into call-by-name equalities, and vice versa.*

$$\begin{array}{lll} M =_v N & \textit{iff} & (M)_\circ\{O\} =_n (N)_\circ\{O\} \\ S =_v T & \textit{iff} & (S)_\circ =_n (T)_\circ, \end{array}$$

with M, N terms of $\lambda\mu$, and S, T statements of $\lambda\mu$.

The proof is easy. For the first line, we have

$$\begin{array}{ll} & M =_v N \\ \text{iff} & (M)^* =_v (N)^* \\ \text{iff} & ((M)^*)^\circ =_n ((N)^*)^\circ \\ \text{iff} & (((M)^*)^\circ)_*\{O\} =_n (((N)^*)^\circ)_*\{O\} \\ \text{iff} & (M)_\circ\{O\} =_n (N)_\circ\{O\} \end{array}$$

The second line is similar.

PROPOSITION 5.7. *(Involution, reloaded) Duality on $\lambda\mu$ is an involution up to equality,*

$$\begin{array}{lll} ((A)_\circ)_\circ & \equiv & A \\ \mu\alpha.\,((M)_\circ\{\bar\alpha\})_\circ & =_v & M \\ ((M)_\circ\{O\})_\circ & =_v & (O)_\circ\{M\} \\ ((S)_\circ)_\circ & =_v & S, \end{array}$$

with A a type, M, O terms of $\lambda\mu$, S a statement of $\lambda\mu$.

This follows from Propositions 4.3, 4.4, and 5.2 We will prove the lines in inverse order. The fourth line is easy,

$$\begin{array}{ll} & ((S)_\circ)_\circ \\ \equiv & ((((((S)^*)^\circ)_*)^*)^\circ)_* \\ =_v & ((((S)^*)^\circ)^\circ)_* \\ \equiv & ((S)^*)_* \\ =_v & S. \end{array}$$

The third line is only slightly harder,

$$\begin{array}{ll} & ((M)_\circ\{O\})_\circ \\ \equiv & ((((((M)^*)^\circ)_*\{O\})^*)^\circ)_* \\ =_v & (((O)^* \bullet ((M)^*)^\circ)^\circ)_* \\ \equiv & ((M)^* \bullet ((O)^*)^\circ)_* \\ \equiv & (((O)^*)^\circ)_*\{((M)^*)_*\} \\ =_v & (O)_\circ\{M\}. \end{array}$$

The second line follows from the third,

$$\begin{array}{ll} & \mu\alpha.\,((M)_\circ\{\bar\alpha\})_\circ \\ =_v & \mu\alpha.\,(\bar\alpha)_\circ\{M\} \\ \equiv & \mu\alpha.\,[\alpha]\,M \\ =_v & M \end{array}$$

The first line is trivial.

Since all of the results of the preceding section hold with $=_v$ replaced by $=_n$, the same holds for the above. However, unlike the preceding section, we don't need to redo any complex case analyses; the additional results follow immediately from the work done previously.

Selinger (2001) gives a duality for $\lambda\mu$ that takes call-by-value into call-by-name, but it is *not* involutive. There are two distinct translations to take call-by-value into call-by-name and call-by-name into call-by-value. Futher, one translation followed by the other does not preserve types up to identity, only up to isomorphism. However, closer inspection shows that the two translations are identical on all components except function types, and agree with the duality translation on $\lambda\mu$ given here. The key difference is that here we have restricted functions in $\lambda\mu$, yielding a cleaner version of duality. Sometimes less is more!

Acknowledgements

Thanks to Pierre-Louis Curien, Olivier Danvy, Ken-etsu Fujita, Vladimir Gapeyev, Tim Griffin, Hugo Herbelin, Paul Levy, Robert McGrail, Rex Page, Bernard Reuss, Peter Selinger, Ken Shan, Thomas Streicher, and Steve Zdancewic for discussions related to this work.

6. REFERENCES

ZENA ARIOLA AND HUGO HERBELIN (2003) Minimal classical logic and control operators. In *30'th International Colloquium on Automata, Languages and Programming*, Eindhoven, The Netherlands.

F. BARBANERA AND S. BERARDI (1996) A symmetric lambda calculus for classical program extraction. *Information and Computation*, 125(2):103–117.

P. BERNAYS (1936) Review of "Some Properties of Conversion" by Alonzo Church and J. B. Rosser. *Journal of Symbolic Logic*, 1:74–75.

ALONZO CHURCH (1932) A set of postulates for the foundation of logic. *Annals of Mathematics*, II.33:346–366.

ALONZO CHURCH (1940) A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68.

P.-L. CURIEN AND H. HERBELIN (2000) The duality of computation. In *5'th International Conference on Functional Programming*, pages 233–243, ACM, September 2000.

V. DANOS, J-B. JOINET AND H. SCHELLINX (1995) LKQ and LKT: Sequent calculi for second order logic based upon linear decomposition of classical implication. In *Advances in Linear Logic*, J-Y. Girard, Y Lafont and L. Regnier editors, London Mathematical Society Lecture Note Series 222, Cambridge University Press, pp. 211-224.

ANDRZEJ FILINSKI (1989) Declarative continuations and categorical duality. Master's thesis, University of Copenhagen, Copenhagen, Denmark, August 1989. (DIKU Report 89/11.)

KEN-ETSU FUJITA (2003) A Sound and Complete CPS-Translation for $\lambda\mu$-Calculus. In Martin Hofmann (editor), *Typed Lambda Calculi and Applications*, Valencia, Spain, 10-12 June 2003, Springer-Verlag, LNCS 2701.

GERHARD GENTZEN (1935) Investigations into Logical Deduction. *Mathematische Zeitschrift* 39:176–210,405–431. Reprinted in M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, North-Holland, 1969.

TIMOTHY GRIFFIN (1990) A formulae-as-types notion of control. In *17'th Symposium on Principles of Programming Languages*, San Francisco, CA, ACM, January 1990.

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

AUSTIN MELTON, BERND S. W. SCHRÖDER, AND GEORGE E. STRECKER (1994) Lagois connections, a counterpart to Galois connections. *Theoretical Computer Science*, 136(1):79–107, 1994.

EUGENIO MOGGI (1988) Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, Edinburgh University, Department of Computer Science.

C.-H. L. ONG (1996) 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.

$$
\begin{array}{lllll}
\text{Type} & A, B & ::= & X \mid A \,\&\, B \mid A \vee B \mid \neg A \\
\text{Term} & M, N, O & ::= & x \mid \langle M, N \rangle \mid \mathrm{fst}\, O \mid \mathrm{snd}\, O \mid \mu[\alpha, \beta].\, S \mid \lambda x.\, S \mid \mu\alpha.\, S \\
\text{Statement} & S, T & ::= & [\alpha]\, M \mid [\alpha, \beta]\, O \mid O\, M \\
\text{Antecedent} & \Gamma & ::= & x_1 : A_1, \ldots, x_m : A_m \\
\text{Succedent} & \Theta & ::= & \beta_1 : B_1, \ldots, \beta_n : B_n
\end{array}
$$

$$
\begin{array}{ll}
\text{Right sequent} & \Gamma \rightarrowtail \Theta \mid M : A \\
\text{Center sequent} & \Gamma \mid S \mapsto \Theta
\end{array}
$$

$$
\frac{}{\Gamma, x : A \rightarrowtail \Theta \mid x : A}\ \mathrm{Id}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A \qquad \Gamma \rightarrowtail \Theta \mid N : B}{\Gamma \rightarrowtail \Theta \mid \langle M, N \rangle : A \,\&\, B}\ \&\mathrm{I}
\qquad
\frac{\Gamma \rightarrowtail \Theta \mid O : A \,\&\, B}{\Gamma \rightarrowtail \Theta \mid \mathrm{fst}\, O : A}
\qquad
\frac{\Gamma \rightarrowtail \Theta \mid O : A \,\&\, B}{\Gamma \rightarrowtail \Theta \mid \mathrm{snd}\, O : B}\ \&\mathrm{E}
$$

$$
\frac{\Gamma \mid S \mapsto \Theta, \alpha : A, \beta : B}{\Gamma \rightarrowtail \Theta \mid \mu[\alpha, \beta].\, S : A \vee B}\ \vee\mathrm{I}
\qquad
\frac{\Gamma \rightarrowtail \Theta, \alpha : A, \beta : B \mid O : A \vee B}{\Gamma \mid [\alpha, \beta]\, O \mapsto \Theta, \alpha : A, \beta : B}\ \vee\mathrm{E}
$$

$$
\frac{\Gamma, x : A \mid S \mapsto \Theta}{\Gamma \rightarrowtail \Theta \mid \lambda x.\, S : \neg A}\ \neg\mathrm{I}
\qquad
\frac{\Gamma \rightarrowtail \Theta \mid O : \neg A \qquad \Gamma \rightarrowtail \Theta \mid M : A}{\Gamma \mid O\, M \mapsto \Theta}\ \neg\mathrm{E}
$$

$$
\frac{\Gamma \mid S \mapsto \Theta, \alpha : A}{\Gamma \rightarrowtail \Theta \mid \mu\alpha.\, S : A}\ \mathrm{Activate}
\qquad
\frac{\Gamma \rightarrowtail \Theta, \alpha : A \mid M : A}{\Gamma \mid [\alpha]\, M \mapsto \Theta, \alpha : A}\ \mathrm{Passivate}
$$

Figure 1: Syntax and types of the $\lambda\mu$-calculus

$$
\begin{array}{lllll}
\text{Term} & M, N & ::= & x \mid \langle M, N \rangle \mid \langle M \rangle \mathrm{inl} \mid \langle N \rangle \mathrm{inr} \mid [K] \mathrm{not} \mid (S).\alpha \\
\text{Coterm} & K, L & ::= & \alpha \mid [K, L] \mid \mathrm{fst}[K] \mid \mathrm{snd}[L] \mid \mathrm{not}\langle M \rangle \mid x.(S) \\
\text{Statement} & S, T & ::= & M \bullet K
\end{array}
$$

$$
\begin{array}{ll}
\text{Right sequent} & \Gamma \rightarrowtail \Theta \mid M : A \\
\text{Left sequent} & K : A \mid \Gamma \rightarrowtail \Theta \\
\text{Center sequent} & \Gamma \mid S \mapsto \Theta
\end{array}
$$

$$
\frac{}{x : A, \Gamma \rightarrowtail \Theta \mid x : A}\ \mathrm{IdR}
\qquad
\frac{}{\alpha : A \mid \Gamma \rightarrowtail \Theta, \alpha : A}\ \mathrm{IdL}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A \qquad \Gamma \rightarrowtail \Theta \mid N : B}{\Gamma \rightarrowtail \Theta \mid \langle M, N \rangle : A \,\&\, B}\ \&\mathrm{R}
\qquad
\frac{K : A \mid \Gamma \rightarrowtail \Theta}{\mathrm{fst}[K] : A \,\&\, B \mid \Gamma \rightarrowtail \Theta}
\qquad
\frac{L : B \mid \Gamma \rightarrowtail \Theta}{\mathrm{snd}[L] : A \,\&\, B \mid \Gamma \rightarrowtail \Theta}\ \&\mathrm{L}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A}{\Gamma \rightarrowtail \Theta \mid \langle M \rangle \mathrm{inl} : A \vee B}
\qquad
\frac{\Gamma \rightarrowtail \Theta \mid N : B}{\Gamma \rightarrowtail \Theta \mid \langle N \rangle \mathrm{inr} : A \vee B}\ \vee\mathrm{R}
\qquad
\frac{K : A \mid \Gamma \rightarrowtail \Theta \qquad L : B \mid \Gamma \rightarrowtail \Theta}{[K, L] : A \vee B \mid \Gamma \rightarrowtail \Theta}\ \vee\mathrm{L}
$$

$$
\frac{K : A \mid \Gamma \rightarrowtail \Theta}{\Gamma \rightarrowtail \Theta \mid [K] \mathrm{not} : \neg A}\ \neg\mathrm{R}
\qquad
\frac{\Gamma \rightarrowtail \Theta \mid M : A}{\mathrm{not}\langle M \rangle : \neg A \mid \Gamma \rightarrowtail \Theta}\ \neg\mathrm{L}
$$

$$
\frac{\Gamma \mid S \mapsto \Theta, \alpha : A}{\Gamma \rightarrowtail \Theta \mid (S).\alpha : A}\ \mathrm{RI}
\qquad
\frac{x : A, \Gamma \mid S \mapsto \Theta}{x.(S) : A \mid \Gamma \rightarrowtail \Theta}\ \mathrm{LI}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A \qquad K : A \mid \Gamma \rightarrowtail \Theta}{\Gamma \mid M \bullet K \mapsto \Theta}\ \mathrm{Cut}
$$

Figure 2: Syntax and types of the dual calculus

$$\text{Values} \quad V, W \quad ::= \quad x \mid \langle V, W\rangle \mid \mu[\alpha,\beta].\,[\alpha]\,V \mid \mu[\alpha,\beta].\,[\beta]\,W \mid \mathrm{fst}\,V \mid \mathrm{fst}\,W$$

$$\begin{array}{lll}
(\beta\&) & \mathrm{fst}\,\langle V, W\rangle & \longrightarrow_v \; V \\
(\beta\&) & \mathrm{snd}\,\langle V, W\rangle & \longrightarrow_v \; W \\
(\beta\vee) & [\alpha,\beta]\,\mu[\alpha',\beta'].\,S & \longrightarrow_v \; S\{\alpha/\alpha', \beta/\beta'\} \\
(\beta\neg) & (\lambda x.\,S)\,V & \longrightarrow_v \; S\{V/x\} \\
(\beta\mu) & [\alpha]\,\mu\alpha'.\,S & \longrightarrow_v \; S\{\alpha'/\alpha\} \\
\\
(\eta\&) & V : A \,\&\, B & \longrightarrow_v \; \langle \mathrm{fst}\,V, \mathrm{snd}\,V\rangle \\
(\eta\vee) & M : A \vee B & \longrightarrow_v \; \mu[\alpha,\beta].\,[\alpha,\beta]\,M \\
(\eta\neg) & V : \neg A & \longrightarrow_v \; \lambda x.\,V\,x \\
(\eta\mu) & M & \longrightarrow_v \; \mu\alpha.\,[\alpha]\,M \\
\\
(\nu\&) & \langle M, N\rangle & \longrightarrow_v \; \mu\gamma.\,(\lambda x.\,[\gamma]\,\langle x, N\rangle)\,M \\
(\nu\&) & \langle V, N\rangle & \longrightarrow_v \; \mu\gamma.\,(\lambda y.\,[\gamma]\,\langle V, y\rangle)\,N \\
(\nu\neg) & O\,M & \longrightarrow_v \; (\lambda z.\,z\,M)\,O \\
\\
(\varsigma\&) & \mathrm{fst}\,\mu\gamma.\,S & \longrightarrow_v \; \mu\alpha.\,S\{[\alpha]\,\mathrm{fst}\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\&) & \mathrm{snd}\,\mu\gamma.\,S & \longrightarrow_v \; \mu\beta.\,S\{[\beta]\,\mathrm{snd}\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\vee) & [\alpha,\beta]\,(\mu\gamma.\,S) & \longrightarrow_v \; S\{[\alpha,\beta]\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\neg) & O\,(\mu\alpha.\,S) & \longrightarrow_v \; S\{O\,\{-\}/[\alpha]\,\{\}\}
\end{array}$$

Figure 3: Reductions of the call-by-value $\lambda\mu$-calculus

$$\begin{array}{lll}
(\beta\&) & \mathrm{fst}\,\langle M, N\rangle & \longrightarrow_n \; M \\
(\beta\&) & \mathrm{snd}\,\langle M, N\rangle & \longrightarrow_n \; N \\
(\beta\vee) & [\alpha,\beta]\,\mu[\alpha',\beta'].\,S & \longrightarrow_n \; S\{\alpha/\alpha', \beta/\beta'\} \\
(\beta\neg) & (\lambda x.\,S)\,M & \longrightarrow_n \; S\{M/x\} \\
(\beta\mu) & [\alpha]\,\mu\alpha'.\,S & \longrightarrow_n \; S\{\alpha'/\alpha\} \\
\\
(\eta\&) & M : A \,\&\, B & \longrightarrow_n \; \langle \mathrm{fst}\,M, \mathrm{snd}\,M\rangle \\
(\eta\vee) & M : A \vee B & \longrightarrow_n \; \mu[\alpha,\beta].\,[\alpha,\beta]\,M \\
(\eta\neg) & M : \neg A & \longrightarrow_n \; \lambda x.\,M\,x \\
(\eta\mu) & M & \longrightarrow_n \; \mu\alpha.\,[\alpha]\,M \\
\\
(\varsigma\vee) & [\alpha,\beta]\,(\mu\gamma.\,S) & \longrightarrow_n \; S\{[\alpha,\beta]\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\&) & \mathrm{fst}\,(\mu\gamma.\,S) & \longrightarrow_n \; \mu\alpha.\,S\{[\alpha]\,\mathrm{fst}\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\&) & \mathrm{snd}\,(\mu\gamma.\,S) & \longrightarrow_n \; \mu\beta.\,S\{[\beta]\,\mathrm{snd}\,\{-\}/[\gamma]\,\{-\}\} \\
(\varsigma\neg) & (\mu\gamma.\,S)\,M & \longrightarrow_n \; S\{\{-\}\,M/[\gamma]\,\{-\}\}
\end{array}$$

Figure 4: Reductions of the call-by-name $\lambda\mu$-calculus

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

M. Parigot (1992) $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In LPAR 1992, pages 190–201, Springer-Verlag, LNCS 624.

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

David Pym and Eike Ritter (2001) On the Semantics of Classical Disjunction. *Journal of Pure and Applied Algebra* 159:315–338.

Amr Sabry and Matthias Felleisen (1993) Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360.

Amr Sabry and Philip Wadler (1997) A reflection on call-by-value. In *ACM Transactions on Programming Languages and Systems*, 19(6):916-941.

Peter Selinger (1998) Control categories and duality: an axiomatic approach to the semantics of functional control. Talk presented at *Mathematical Foundations of Programming Semantics*, London, May 1998.

Peter Selinger (2001) Control categories and duality: on the categorical semantics of the lambda-mu calculus. In *Mathematical Structures in Computer Science*, 11:207–260.

Thomas Streicher and Bernard Reuss (1998)

Value $V, W ::= x \mid \langle V, W\rangle \mid \langle V\rangle\mathrm{inl} \mid \langle W\rangle\mathrm{inr} \mid [K]\mathrm{not} \mid \lambda x.\, N$

$(\beta\&)$	$\langle V, W\rangle \bullet \mathrm{fst}[K]$	$\longrightarrow^v$	$V \bullet K$
$(\beta\&)$	$\langle V, W\rangle \bullet \mathrm{snd}[L]$	$\longrightarrow^v$	$W \bullet L$
$(\beta\vee)$	$\langle V\rangle\mathrm{inl} \bullet [K, L]$	$\longrightarrow^v$	$V \bullet K$
$(\beta\vee)$	$\langle W\rangle\mathrm{inr} \bullet [K, L]$	$\longrightarrow^v$	$W \bullet L$
$(\beta\neg)$	$[K]\mathrm{not} \bullet \mathrm{not}\langle M\rangle$	$\longrightarrow^v$	$M \bullet K$
$(\beta\mathrm{L})$	$V \bullet x.(S)$	$\longrightarrow^v$	$S\{V/x\}$
$(\beta\mathrm{R})$	$(S).\alpha \bullet K$	$\longrightarrow^v$	$S\{K/\alpha\}$
$(\eta\&)$	$V : A \,\&\, B$	$\longrightarrow^v$	$\langle (V \bullet \mathrm{fst}[\alpha]).\alpha, (V \bullet \mathrm{snd}[\beta]).\beta\rangle$
$(\eta\vee)$	$K : A \vee B$	$\longrightarrow^v$	$[x.(\langle x\rangle\mathrm{inl} \bullet K), y.(\langle y\rangle\mathrm{inr} \bullet K)]$
$(\eta\neg)$	$V : \neg A$	$\longrightarrow^v$	$[x.(V \bullet \mathrm{not}\langle x\rangle)]\mathrm{not}$
$(\eta\neg)$	$K : \neg A$	$\longrightarrow^v$	$\mathrm{not}\langle([\alpha]\mathrm{not} \bullet K).\alpha\rangle$
$(\eta\mathrm{L})$	K	$\longrightarrow^v$	$x.(x \bullet K)$
$(\eta\mathrm{R})$	M	$\longrightarrow^v$	$(M \bullet \alpha).\alpha$
$(\nu\&)$	$\langle M, N\rangle \bullet K$	$\longrightarrow^v$	$M \bullet x.(\langle x, N\rangle \bullet K)$
$(\nu\&)$	$\langle V, N\rangle \bullet K$	$\longrightarrow^v$	$N \bullet y.(\langle V, y\rangle \bullet K)$
$(\nu\vee)$	$\langle M\rangle\mathrm{inl} \bullet K$	$\longrightarrow^v$	$M \bullet x.(\langle x\rangle\mathrm{inl} \bullet K)$
$(\nu\vee)$	$\langle N\rangle\mathrm{inr} \bullet K$	$\longrightarrow^v$	$N \bullet y.(\langle y\rangle\mathrm{inr} \bullet K)$
$(\varsigma\&)$	$V \bullet \mathrm{fst}[x.(S)]$	$\longrightarrow^v$	$S\{(V \bullet \mathrm{fst}[\alpha]).\alpha/x\}$
$(\varsigma\&)$	$V \bullet \mathrm{snd}[y.(S)]$	$\longrightarrow^v$	$S\{(V \bullet \mathrm{snd}[\beta]).\beta/y\}$

Figure 5: Reductions of the call-by-value dual calculus

Covalue $P, Q ::= \alpha \mid [P, Q] \mid \mathrm{fst}[P] \mid \mathrm{snd}[Q] \mid \mathrm{not}\langle M\rangle \mid M \mathbin{@} Q$

$(\beta\vee)$	$\langle M\rangle\mathrm{inl} \bullet [P, Q]$	$\longrightarrow^n$	$M \bullet P$
$(\beta\vee)$	$\langle N\rangle\mathrm{inr} \bullet [P, Q]$	$\longrightarrow^n$	$N \bullet Q$
$(\beta\&)$	$\langle M, N\rangle \bullet \mathrm{fst}[P]$	$\longrightarrow^n$	$M \bullet P$
$(\beta\&)$	$\langle M, N\rangle \bullet \mathrm{snd}[Q]$	$\longrightarrow^n$	$N \bullet Q$
$(\beta\neg)$	$[K]\mathrm{not} \bullet \mathrm{not}\langle M\rangle$	$\longrightarrow^n$	$M \bullet K$
$(\beta\mathrm{R})$	$(S).\alpha \bullet P$	$\longrightarrow^n$	$S\{P/\alpha\}$
$(\beta\mathrm{L})$	$M \bullet x.(S)$	$\longrightarrow^n$	$S\{M/x\}$
$(\eta\vee)$	$P : A \vee B$	$\longrightarrow^n$	$[x.(\langle x\rangle\mathrm{inl} \bullet P), y.(\langle y\rangle\mathrm{inr} \bullet P)]$
$(\eta\&)$	$M : A \,\&\, B$	$\longrightarrow^n$	$\langle (M \bullet \mathrm{fst}[\alpha]).\alpha, (M \bullet \mathrm{snd}[\beta]).\beta\rangle$
$(\eta\neg)$	$P : \neg A$	$\longrightarrow^n$	$\mathrm{not}\langle([\alpha]\mathrm{not} \bullet P).\alpha\rangle$
$(\eta\neg)$	$M : \neg A$	$\longrightarrow^n$	$[x.(M \bullet \mathrm{not}\langle x\rangle)]\mathrm{not}$
$(\eta\mathrm{R})$	M	$\longrightarrow^n$	$(M \bullet \alpha).\alpha$
$(\eta\mathrm{L})$	K	$\longrightarrow^n$	$x.(x \bullet K)$
$(\nu\vee)$	$M \bullet [K, L]$	$\longrightarrow^n$	$(M \bullet [\alpha, L]).\alpha \bullet K$
$(\nu\vee)$	$M \bullet [P, L]$	$\longrightarrow^n$	$(M \bullet [P, \beta]).\beta \bullet L$
$(\nu\&)$	$M \bullet \mathrm{fst}[K]$	$\longrightarrow^n$	$(M \bullet \mathrm{fst}[\alpha]).\alpha \bullet K$
$(\nu\&)$	$M \bullet \mathrm{snd}[L]$	$\longrightarrow^n$	$(M \bullet \mathrm{snd}[\beta]).\beta \bullet L$
$(\varsigma\vee)$	$\langle (S).\alpha\rangle\mathrm{inl} \bullet P$	$\longrightarrow^n$	$S\{x.(\langle x\rangle\mathrm{inl} \bullet P)/\alpha\}$
$(\varsigma\vee)$	$\langle (S).\beta\rangle\mathrm{inr} \bullet P$	$\longrightarrow^n$	$S\{y.(\langle y\rangle\mathrm{inr} \bullet P)/\beta\}$

Figure 6: Reductions of the call-by-name dual calculus

$$(M)^* \equiv ((M)^*\{\alpha\}).\alpha$$

$$\begin{array}{lcl}
(x)^*\{P\} & \equiv & x \bullet P \\
(\langle M, N\rangle)^*\{P\} & \equiv & \langle (M)^*, (N)^*\rangle \bullet P \\
(\mathrm{fst}\, O)^*\{P\} & \equiv & (O)^*\{\mathrm{fst}[P]\} \\
(\mathrm{snd}\, O)^*\{P\} & \equiv & (O)^*\{\mathrm{snd}[P]\} \\
(\mu[\alpha,\beta].\, S)^*\{P\} & \equiv & \langle(\langle((S)^*).\beta\rangle \mathrm{inr} \bullet P).\alpha\rangle \mathrm{inl} \bullet P \\
(\lambda x.\, S)^*\{P\} & \equiv & [x.((S)^*)]\mathrm{not} \bullet P \\
(\mu\alpha.\, S)^*\{P\} & \equiv & ((S)^*).\alpha \bullet P
\end{array}
\qquad
\begin{array}{lcl}
([\alpha,\beta]\, O)^* & \equiv & (O)^*\{[\alpha,\beta]\} \\
(O\, M)^* & \equiv & (O)^*\{\mathrm{not}\langle (M)^*\rangle\} \\
([\alpha]\, M)^* & \equiv & (M)^*\{\alpha\}
\end{array}$$

Figure 7: Translation from $\lambda\mu$-calculus to dual calculus

$$\begin{array}{lcl}
(x)_* & \equiv & x \\
(\langle M, N\rangle)_* & \equiv & \langle (M)_*, (N)_*\rangle \\
(\langle M\rangle \mathrm{inl})_* & \equiv & \mu[\alpha,\beta].\, [\alpha]\, (M)_* \\
(\langle N\rangle \mathrm{inr})_* & \equiv & \mu[\alpha,\beta].\, [\beta]\, (N)_* \\
([K]\mathrm{not})_* & \equiv & \lambda x.\, (K)_*\{x\} \\
((S).\alpha)_* & \equiv & \mu\alpha.\, (S)_*
\end{array}
\qquad
\begin{array}{lcl}
(\alpha)_*\{O\} & \equiv & [\alpha]\, O \\
([K, L])_*\{O\} & \equiv & (K)_*\{\mu\alpha.\, (L)_*\{\mu\beta.\, [\alpha,\beta]\, O\}\} \\
(\mathrm{fst}[K])_*\{O\} & \equiv & (K)_*\{\mathrm{fst}\, O\} \\
(\mathrm{snd}[L])_*\{O\} & \equiv & (L)_*\{\mathrm{snd}\, O\} \\
(\mathrm{not}\langle M\rangle)_*\{O\} & \equiv & O\, (M)_* \\
(x.(S))_*\{O\} & \equiv & (\lambda x.\, (S)_*)\, O
\end{array}$$

$$(M \bullet K)_* \equiv (K)_*\{(M)_*\}$$

Figure 8: Translation from dual calculus to $\lambda\mu$-calculus

$$\begin{array}{lcl}
(X)^\circ & \equiv & X \\
(A \,\&\, B)^\circ & \equiv & (A)^\circ \vee (B)^\circ \\
(A \vee B)^\circ & \equiv & (A)^\circ \,\&\, (B)^\circ \\
(\neg A)^\circ & \equiv & \neg(A)^\circ
\end{array}$$

$$\begin{array}{lcl}
(x)^\circ & \equiv & \bar{x} \\
(\langle M, N\rangle)^\circ & \equiv & [(M)^\circ, (N)^\circ] \\
(\langle M\rangle \mathrm{inl})^\circ & \equiv & \mathrm{fst}[(M)^\circ] \\
(\langle N\rangle \mathrm{inr})^\circ & \equiv & \mathrm{snd}[(M)^\circ] \\
([K]\mathrm{not})^\circ & \equiv & \mathrm{not}\langle (K)^\circ\rangle \\
((S).\alpha)^\circ & \equiv & \bar{\alpha}.((S)^\circ)
\end{array}
\qquad
\begin{array}{lcl}
(\alpha)^\circ & \equiv & \bar{\alpha} \\
([K, L])^\circ & \equiv & \langle (K)^\circ, (L)^\circ\rangle \\
(\mathrm{fst}[K])^\circ & \equiv & \langle (K)^\circ\rangle \mathrm{inl} \\
(\mathrm{snd}[L])^\circ & \equiv & \langle (K)^\circ\rangle \mathrm{inr} \\
(\mathrm{not}\langle M\rangle)^\circ & \equiv & [(M)^\circ]\mathrm{not} \\
(x.(S))^\circ & \equiv & ((S)^\circ).\bar{x}
\end{array}$$

$$(M \bullet K)^\circ \equiv (K)^\circ \bullet (M)^\circ$$

Figure 9: Duality for the dual calculus

$$(A)_\circ \equiv (A)^\circ$$

$$\begin{array}{lcl}
(x)_\circ\{O'\} & \equiv & [\bar{x}]\, O' \\
(\langle M, N\rangle)_\circ\{O'\} & \equiv & (M)_\circ\{\mu\alpha.\, (N)_\circ\{\mu\beta.\, [\alpha,\beta]\, O'\}\} \\
(\mathrm{fst}\, O)_\circ\{O'\} & \equiv & (\lambda x.\, (O)_\circ\{\mu[\alpha,\beta].\, [\alpha]\, x\})\, O' \\
(\mathrm{snd}\, O)_\circ\{O'\} & \equiv & \lambda y.\, (O)_\circ\{\mu[\alpha,\beta].\, [\beta]\, y\}\, O' \\
(\mu[\alpha,\beta].\, S)_\circ\{O'\} & \equiv & (\lambda z.\, (\lambda\bar{\alpha}.\, (\lambda\bar{\beta}.\, (S)_\circ)\, (\mathrm{snd}[z]))\, (\mathrm{fst}[z]))\, O' \\
(\lambda x.\, S)_\circ\{O'\} & \equiv & O'\, (\mu\bar{x}.\, (S)_\circ) \\
(\mu\alpha.\, S)_\circ\{O'\} & \equiv & (\lambda\bar{\alpha}.\, (S)_\circ)\, O'
\end{array}
\qquad
\begin{array}{lcl}
([\alpha,\beta]\, O)_\circ & \equiv & (O)_\circ\{\langle\bar{\alpha},\bar{\beta}\rangle\} \\
(O\, M)_\circ & \equiv & (O)_\circ\{\lambda x.\, (M)_\circ\{x\}\} \\
([\alpha]\, M)_\circ & \equiv & (M)_\circ\{\bar{\alpha}\}
\end{array}$$

Figure 10: Duality for the $\lambda\mu$-calculus

Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.

Philip Wadler (2003) Call-by-name is dual to call-by-value. In *International Conference on Functional Programming*, Uppsala, Sweden, 25–29 August 2003.