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

Philip Wadler

Edinburgh University

Abstract. We consider the relation of the dual calculus of Wadler (2003) to the $\lambda\mu$-calculus of Parigot (1992). We give translations from the $\lambda\mu$-calculus into the dual calculus and back again. The translations form an equational correspondence as defined by Sabry and Felleisen (1993). In particular, 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://homepages.inf.ed.ac.uk/wadler/

or google 'wadler dual reloaded'.

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 $A \supset B \equiv \neg A \vee B$. 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 an implication $A \supset B$ is the difference operator, $B - A \equiv B \,\&\, \neg A$, 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

---

* Invited talk, *Rewriting Techniques and Applications*, Nara, April 2005. (Revised,

May 2008.)

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

$$M =_v N, \quad M =^v N$$

for equality in the source and target respectively, and

$$M^*, \quad M_*$$

for translations from source to target and target to source respectively. 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) also presents a CPS translation from the dual calculus into λ-calculus, again in both call-by-value and call-by-name variants. Composing

the translation from the $\lambda\mu$-calculus to the dual calculus with the CPS translation for the dual calculus 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).

Following the technique introduced in Sabry and Wadler (1997), it is shown that the CPS translation for the dual calculus is a *reflection*, that is it both preserves and reflects reductions. Every reflection is trivially an equational correspondence, where equality is the reflexive, symmetric, and transitive closure of reduction. Since equational correspondences compose, it follows immediately that the CPS translation for $\lambda\mu$-calculus is also an equational correspondence.

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 call-by-value dual calculus into the call-by-name dual calculus, and conversely; that is, if two terms are equal in the call-by-value calculus then their duals are equal call-by-name. Duality is an involution; that is, the dual of the dual is the identity. It follows immediately that duality is an equational correspondence.

Our type system corresponds to minimal logic, with types $A \,\&\, B$, $A \vee B$, $\neg A$, and $A \supset B$ corresponding to 'and', 'or', 'not', and 'implies'. (We would

have $\neg A = A \supset \bot$, if we had defined a type $\bot$ corresponding to 'false'.) Duality exchanges 'and' with 'or', and 'not' is self dual. The dual of implication $A \supset B = \neg A \vee B$ is difference $B - A = B \,\&\, \neg A$. (One can confirm this by checking $B - A = \neg(\neg A \supset \neg B)$.) We choose not to include difference in our type system, because its computational interpretation is not familiar. (For one exploration of what the computational interpretation of difference might be, see Crolard (2004).) It follows that before we consider duality, we first must translate away implications. We use the translation $A \supset B = \neg(A \,\&\, \neg B)$ for call-by-value and $A \supset B = \neg A \vee B$ for call-by-name.

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. Selinger's duality required some cleverness to construct — it answered an open question of Streicher and Reuss (1998).

As one would hope, Selinger's duality is an involution for the types corresponding to 'and' and 'or'. However, Selinger has no type corresponding directly to 'not', so he is forced to consider what the dual of an implication might be. Since he has no type corresponding to difference, he is forced to require two distinct mappings, one from call-by-value into call-by-name and one from call-

by-name into call-by-value. Further, the composition of these maps does not yield the identity but only the identity up to isomorphism of types. Here we avoid the problem by adding a negation type to the $\lambda\mu$-calculus, requiring that one translate implications before computing the dual. The result is that for us duality on $\lambda\mu$ becomes a proper involution.

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.

Wadler (2003) considers reductions, while this paper considers equations. One advantage of considering equations is that it is then easy to add (η) rules, which are problematic for reductions in the presence of sums (see Balat, di Cosmo, and Fiore (2004)). An interesting open question is whether one can replace the equations of this paper by reductions (possibly omitting the (η) rules), and refine the equational correspondence to a reflection.

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. We provide two variants of λ-abstraction, one where the body is a statement (corresponding to negation), and one where the body is an expressions (corresponding to implication). Informally, one can think of these as related by the equation $\neg A = A \supset \bot$.

Let A, B range over types. A type is atomic X; a conjunction $A \,\&\, B$; a disjunction $A \vee B$; a negation $\neg A$; or an implication $A \supset B$.

Let x, y, z range over variables, α, β, γ range over covariables, 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 $\lambda x.\, N$; a negation application $O\, M$ (where $O : \neg A$); or a μ-abstration $\mu\alpha.\, S$. A statement is a function application $O\, M$ (where $O : A \supset B$); 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

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

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

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

$$
\frac{\Gamma \rightharpoonup \Theta \mid M : A \qquad \Gamma \rightharpoonup \Theta \mid N : B}{\Gamma \rightharpoonup \Theta \mid \langle M, N \rangle : A \,\&\, B}\ \&\mathrm{I}
$$

$$
\frac{\Gamma \rightharpoonup \Theta \mid O : A \,\&\, B}{\Gamma \rightharpoonup \Theta \mid \mathrm{fst}\, O : A}\ \&\mathrm{E} \qquad \frac{\Gamma \rightharpoonup \Theta \mid O : A \,\&\, B}{\Gamma \rightharpoonup \Theta \mid \mathrm{snd}\, O : B}\ \&\mathrm{E}
$$

$$
\frac{\Gamma \mid S \vdash \Theta, \alpha : A, \beta : B}{\Gamma \rightharpoonup \Theta \mid \mu[\alpha, \beta].\, S : A \vee B}\ \vee\mathrm{I}
$$

$$\frac{\Gamma \rightharpoonup \Theta, \alpha : A, \beta : B \mid O : A \vee B}{\Gamma \mid [\alpha, \beta]O \vdash\!\!\rightharpoonup \Theta, \alpha : A, \beta : B} \vee\mathrm{E}$$

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

$$\frac{x : A, \Gamma \rightharpoonup \Theta \mid N : B}{\Gamma \rightharpoonup \Theta \mid \lambda x.\, B : A \supset B} \supset\mathrm{I} \qquad \frac{\Gamma \rightharpoonup \Theta \mid O : A \supset B \qquad \Gamma \rightharpoonup \Theta \mid M : A}{\Gamma \rightharpoonup \Theta \mid O\, M : B} \supset\mathrm{E}$$

$$\frac{\Gamma \mid S \vdash\!\!\rightharpoonup \Theta, \alpha : A}{\Gamma \rightharpoonup \Theta \mid \mu\alpha.\, S : A}\ \text{Activate} \qquad \frac{\Gamma \rightharpoonup \Theta, \alpha : A \mid M : A}{\Gamma \mid [\alpha]M \vdash\!\!\rightharpoonup \Theta, \alpha : A}\ \text{Passivate}$$

Fig. 1. Syntax and types of the $\lambda\mu$-calculus

Values	V, W	$::= x \mid \langle V, W\rangle \mid \mu[\alpha, \beta].\,[\alpha]V \mid \mu[\alpha, \beta].\,[\beta]W \mid \lambda x.\, S \mid \lambda x.\, N \mid \mathrm{fst}\, V \mid \mathrm{fst}\, W$
Evaluation context	E	$::= \{-\} \mid \langle E, N\rangle \mid \langle V, E\rangle \mid \mathrm{fst}\, E \mid \mathrm{snd}\, E \mid E\, M \mid V\, E$
Statement context	D	$::= [\alpha]E \mid [\alpha, \beta]E \mid E\, M \mid V\, E$

$$\begin{array}{lll}
(\beta\&) & \mathrm{fst}\,\langle V, W\rangle & =_v V \\
(\beta\&) & \mathrm{snd}\,\langle V, W\rangle & =_v W \\
(\beta\vee) & [\alpha, \beta]\mu[\alpha', \beta'].\, S & =_v S\{\alpha/\alpha', \beta/\beta'\} \\
(\beta\neg) & (\lambda x.\, S)\, V & =_v S\{V/x\}
\end{array}$$

$$
\begin{array}{lll}
(\beta\supset) & (\lambda x.\, N)\, V & =_v N\{V/x\} \\
(\beta\mu) & [\alpha]\mu\alpha'.\, S & =_v S\{\alpha'/\alpha\} \\
 & & \\
(\eta\&) & V : A \,\&\, B & =_v \langle \mathrm{fst}\, V, \mathrm{snd}\, V\rangle \\
(\eta\vee) & M : A \vee B & =_v \mu[\alpha,\beta].\, [\alpha,\beta]M \\
(\eta\neg) & V : \neg A & =_v \lambda x.\, V\, x \\
(\eta\supset) & V : A \supset B & =_v \lambda x.\, V\, x \\
(\eta\mu) & M & =_v \mu\alpha.\, [\alpha]M \\
 & & \\
(\mathrm{name}) & D\{M\} & =_v (\lambda x.\, D\{x\})\, M \\
(\mathrm{comp}) & D\{(\lambda x.\, N)\, M\} & =_v (\lambda x.\, D\{N\})\, M \\
(\varsigma) & D\{\mu\alpha.\, S\} & =_v S\{D\{-\}/[\alpha]\{-\}\}
\end{array}
$$

Fig. 2. Equations of the call-by-value $\lambda\mu$-calculus

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

$$
\begin{array}{lll}
(\eta\neg) & M : \neg A & =_n \lambda x.\, M\, x \\
(\eta\supset) & M : A \supset B & =_n \lambda x.\, M\, x \\
(\eta\mu) & M & =_n \mu\alpha.\, [\alpha]M \\
 & & \\
(\varsigma\vee) & [\alpha,\beta](\mu\gamma.\, S) & =_n S\{[\alpha,\beta]\{-\}/[\gamma]\{-\}\} \\
(\varsigma\&) & \mathrm{fst}\ (\mu\gamma.\, S) & =_n \mu\alpha.\, S\{[\alpha]\mathrm{fst}\ \{-\}/[\gamma]\{-\}\} \\
(\varsigma\&) & \mathrm{snd}\ (\mu\gamma.\, S) & =_n \mu\beta.\, S\{[\beta]\mathrm{snd}\ \{-\}/[\gamma]\{-\}\} \\
(\varsigma\neg) & (\mu\gamma.\, S)\, M & =_n S\{\{-\}\, M/[\gamma]\{-\}\} \\
(\varsigma\supset) & (\mu\gamma.\, S)\, M & =_n \mu\beta.\, S\{[\beta]\{-\}\, M/[\gamma]\{-\}\}
\end{array}
$$

Fig. 3. Equations of the call-by-name $\lambda\mu$-calculus

(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 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 equalities, written $=_v$ are shown in Figure 3, and call-by-name equalities, written $=_n$ are shown in Figure 4.

For the call-by-value calculus we need a notion of value, and notions of evaluation and statement contexts. Let V, W range over values, E range over evaluation contexts, and D range over statement contexts. A value is a variable, a pair of values, an injection of a value, a function, or a projection from a value. An evaluation context is a term with a hole, and a statement context is a statement with a hole, such that any term substituted into the hole will be the next to be evaluated. We write $\{-\}$ for the hole; the result of placing term M into the hole in an evaluation context E is written $E\{M\}$, similarly for statement contexts.

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; and some additional rules. In the call-by-value calculus, three rules are stated with statement contexts. It is easy to prove, using $(\eta\mu)$, that the rules also hold when the statement context D is replaced with an evaluation context E. The (name) rule introduces a name for the next term to be evaluated; it is similar to

the rules (let.1) and (let.2) in the λ_c-calculus of Moggi (1988) and the various (let) rules in Selinger (2001). The (comp) rule is similar to the associativity rule in the the λ_c-calculus of Moggi (1988), and the (let) rule in Selinger (2001).

$$\begin{array}{lll}
\text{(let.1)} & O\,M & =_v \text{ let } z = O \text{ in } z\,N \\
\text{(let.2)} & V\,M & =_v \text{ let } x = M \text{ in } V\,x \\
\text{(comp)} & \text{let } y = (\text{let } x = M \text{ in } N) \text{ in } O & =_v \text{ let } x = M \text{ in let } y = N \text{ in } O
\end{array}$$

The (ς) rules of the call-by-value and call-by-name calculi are similar to the (ς) rules of Selinger (2001).

As noted, implication can be defined in terms of the other connectives, but different definitions must be used for call-by-value or call-by-name.

Proposition 1. *Under call-by-value, implication may be defined by*

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

validating ($\beta\supset$), ($\eta\supset$), and the other equations for functions, and where the translation of a function abstraction is a value.

Proposition 2. *Under call-by-name, implication may be defined by*

$$A \supset B \equiv \neg A \vee B$$

$$\begin{aligned} \lambda x.\, N &\equiv \mu[\gamma,\beta].\,[\gamma]\lambda x.\,[\beta]N \\ O\; M &\equiv \mu\beta.\,(\mu\gamma.\,[\gamma,\beta]O)\; M \end{aligned}$$

validating $(\beta\supset)$, $(\eta\supset)$, *and* $(\varsigma\supset)$.

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.

$$(\beta\mathrm{L}) \qquad M \bullet x.(S) = S\{M/x\}$$
$$(\beta\mathrm{R}) \qquad (S).\alpha \bullet K = S\{K/\alpha\}$$

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 $S\{K/\alpha\}$.

Type	A, B	$::= X \mid A \,\&\, B \mid A \vee B \mid \neg A \mid A \supset B$
Term	M, N	$::= x \mid \langle M, N\rangle \mid \langle M\rangle\mathrm{inl} \mid \langle N\rangle\mathrm{inr} \mid [K]\mathrm{not} \mid \lambda x.N \mid (S).\alpha$
Coterm	K, L	$::= \alpha \mid [K, L] \mid \mathrm{fst}[K] \mid \mathrm{snd}[L] \mid \mathrm{not}\langle M\rangle \mid M \mathbin{@} L \mid x.(S)$
Statement	S, T	$::= M \bullet K$
Antecedent	Γ	$::= x_1 : A_1, \ldots, x_m : A_m$
Succedent	Θ	$::= \beta_1 : B_1, \ldots, \beta_n : B_n$

Right sequent	$\Gamma \to \Theta \mid M : A$
Left sequent	$K : A \mid \Gamma \to \Theta$
Center sequent	$\Gamma \mid S \mapsto \Theta$

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

$$\frac{\Gamma \to \Theta \mid M : A \qquad \Gamma \to \Theta \mid N : B}{\Gamma \to \Theta \mid \langle M, N \rangle : A \,\&\, B}\ \&\mathrm{R}$$

$$\frac{K : A \mid \Gamma \to \Theta}{\mathrm{fst}[K] : A \,\&\, B \mid \Gamma \to \Theta}\ \&\mathrm{L} \qquad \frac{L : B \mid \Gamma \to \Theta}{\mathrm{snd}[L] : A \,\&\, B \mid \Gamma \to \Theta}\ \&\mathrm{L}$$

$$\frac{\Gamma \to \Theta \mid M : A}{\Gamma \to \Theta \mid \langle M \rangle \mathrm{inl} : A \vee B}\ \vee\mathrm{R} \qquad \frac{\Gamma \to \Theta \mid N : B}{\Gamma \to \Theta \mid \langle N \rangle \mathrm{inr} : A \vee B}\ \vee\mathrm{R}$$

$$\frac{K : A \mid \Gamma \to \Theta \qquad L : B \mid \Gamma \to \Theta}{[K, L] : A \vee B \mid \Gamma \to \Theta}\ \vee\mathrm{L}$$

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

$$\frac{x : A, \Gamma \to \Theta \mid N : B}{\Gamma \to \Theta \mid \lambda x.N : A \supset B}\ \supset\mathrm{R} \qquad \frac{\Gamma \to \Theta \mid M : A \qquad L : B \mid \Gamma \to \Theta}{M \mathbin{@} L : A \supset B \mid \Gamma \to \Theta}\ \supset\mathrm{L}$$

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

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

Fig. 4. Syntax and types of the dual calculus

Value		V, W	$::= x \mid \langle V, W\rangle \mid \langle V\rangle\text{inl} \mid \langle W\rangle\text{inr} \mid [K]\text{not} \mid \lambda x.N \mid (V \bullet \text{fst}[\alpha]).\alpha \mid (V \bullet \text{snd}[\beta]).\beta$
Evaluation context		E	$::= \{-\} \mid \langle E, N\rangle \mid \langle V, E\rangle \mid \langle E\rangle\text{inl} \mid \langle E\rangle\text{inr}$
$(\beta\&)$	$\langle V, W\rangle \bullet \text{fst}[K]$	$=^v$	$V \bullet K$
$(\beta\&)$	$\langle V, W\rangle \bullet \text{snd}[L]$	$=^v$	$W \bullet L$
$(\beta\vee)$	$\langle V\rangle\text{inl} \bullet [K, L]$	$=^v$	$V \bullet K$
$(\beta\vee)$	$\langle W\rangle\text{inr} \bullet [K, L]$	$=^v$	$W \bullet L$
$(\beta\neg)$	$[K]\text{not} \bullet \text{not}\langle M\rangle$	$=^v$	$M \bullet K$
$(\beta\supset)$	$\lambda x.N \bullet M \mathbin{@} L$	$=^v$	$M \bullet x.(N \bullet L)$
(βL)	$V \bullet x.(S)$	$=^v$	$S\{V/x\}$
(βR)	$(S).\alpha \bullet K$	$=^v$	$S\{K/\alpha\}$
$(\eta\&)$	$V : A \,\&\, B$	$=^v$	$\langle (V \bullet \text{fst}[\alpha]).\alpha, (V \bullet \text{snd}[\beta]).\beta\rangle$
$(\eta\vee)$	$K : A \vee B$	$=^v$	$[x.(\langle x\rangle\text{inl} \bullet K), y.(\langle y\rangle\text{inr} \bullet K)]$
$(\eta\neg)$	$V : \neg A$	$=^v$	$[x.(V \bullet \text{not}\langle x\rangle)]\text{not}$
$(\eta\supset)$	$V : A \supset B$	$=^v$	$\lambda x.((V \bullet x \mathbin{@} \beta).\beta)$
(ηL)	K	$=^v$	$x.(x \bullet K)$
(ηR)	M	$=^v$	$(M \bullet \alpha).\alpha$

$$\text{(name)} \qquad E\{M\} \bullet K \qquad =^{v} \; M \bullet x.(E\{x\} \bullet K)$$

Fig. 5. Equations of the call-by-value dual calculus

$$\begin{array}{lll}
\text{Covalue} & P, Q ::= & \alpha \mid [Q, P] \mid \mathrm{snd}[P] \mid \mathrm{fst}[Q] \mid \mathrm{not}\langle M\rangle \mid \\
& & M \mathbin{@} Q \mid x.(\langle x\rangle \mathrm{inr} \bullet P) \mid y.(\langle y\rangle \mathrm{inl} \bullet P) \\
\text{Coevaluation context} & F \quad ::= & \{-\} \mid [L, F] \mid [F, P] \mid \mathrm{snd}[F] \mid \mathrm{fst}[F]
\end{array}$$

$$\begin{array}{lll}
(\beta\vee) & \langle M\rangle \mathrm{inr} \bullet [Q, P] & =^{n} \; M \bullet P \\
(\beta\vee) & \langle N\rangle \mathrm{inl} \bullet [Q, P] & =^{n} \; N \bullet Q \\
(\beta\&) & \langle N, M\rangle \bullet \mathrm{snd}[P] & =^{n} \; M \bullet P \\
(\beta\&) & \langle N, M\rangle \bullet \mathrm{fst}[Q] & =^{n} \; N \bullet Q \\
(\beta\neg) & [K]\mathrm{not} \bullet \mathrm{not}\langle M\rangle & =^{n} \; M \bullet K \\
(\beta\supset) & \lambda x.N \bullet M \mathbin{@} L & =^{n} \; M \bullet x.(N \bullet L) \\
(\beta\mathrm{R}) & (S).\alpha \bullet P & =^{n} \; S\{P/\alpha\} \\
(\beta\mathrm{L}) & M \bullet x.(S) & =^{n} \; S\{M/x\}
\end{array}$$

$$\begin{array}{lll}
(\eta\vee) & P : A \vee B & =^{n} \; [y.(\langle y\rangle \mathrm{inl} \bullet P), x.(\langle x\rangle \mathrm{inr} \bullet P)] \\
(\eta\&) & M : A \,\&\, B & =^{n} \; \langle (M \bullet \mathrm{fst}[\beta]).\beta, (M \bullet \mathrm{snd}[\alpha]).\alpha\rangle \\
(\eta\neg) & P : \neg A & =^{n} \; \mathrm{not}\langle([\alpha]\mathrm{not} \bullet P).\alpha\rangle \\
(\eta\supset) & M : A \supset B & =^{n} \; \lambda x.((M \bullet x \mathbin{@} \beta).\beta) \\
(\eta\mathrm{R}) & M & =^{n} \; (M \bullet \alpha).\alpha \\
(\eta\mathrm{L}) & K & =^{n} \; x.(x \bullet K)
\end{array}$$

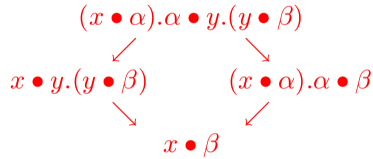
$$\text{(name)} \qquad M \bullet F\{K\} \qquad =^n (M \bullet F\{\alpha\}).\alpha \bullet K$$

Fig. 6. Equations of the call-by-name dual calculus

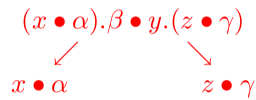
A critical pair occurs when a covariable abstraction is cut against a variable abstraction.

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

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.

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

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.

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

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 equalities, written $=^{v}$, are shown in Figure 5 and call-by-name equalities, written $=^{n}$, are shown in Figure 6.

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

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, any complement, an

application context where the second component is a covalue, or a left or right injection into a covalue. 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), $(\eta,)$ (name) and (ς) rules. The (name) rules correspond to the (ς) rules of Wadler (2003).

As before, implication can be defined in terms of the other connectives, but different definitions must be used for call-by-value or call-by-name. Under call-by-value function abstractions must translate to values, while under call-by-name function applications must translate to covalues, and this is what forces different definitions for the two reduction disciplines.

Proposition 3. *Under call-by-value, implication may be defined by*

$$\begin{array}{lll} A \supset B & \equiv & \neg(A \,\&\, \neg B) \\ \lambda x.N & \equiv & [z.(z \bullet fst[x.(z \bullet snd[not\langle N\rangle])])]not \\ M \mathbin{@} L & \equiv & not\langle\langle M, [L]not\rangle\rangle. \end{array}$$

validating $(\beta\supset)$, $(\eta\supset)$, and the other equations for functions, and where the translation of a function abstraction is a value.

Proposition 4. *Under call-by-name, implication can be defined by*

$$\begin{array}{lll} A \supset B & \equiv & \neg A \vee B \\ \lambda x.N & \equiv & (\langle[x.(\langle N\rangle inr \bullet \gamma)]not\rangle inl \bullet \gamma).\gamma \end{array}$$

$$M \mathbin{@} L \equiv [not\langle M\rangle, L].$$

validating $(\beta\supset)$, $(\eta\supset)$, *and the other equations for functions, and where the translation of a function application is a covalue.*

4 Translations

We now consider the translation from the $\lambda\mu$-calculus to the dual calculus and its inverse translation. The results of this section apply to all types, including implication.

Definition 2. *The translation from the* $\lambda\mu$*-calculus into the dual calculus is given in Figure 7. It consists of two operations,*

$$M^*, \quad S^*.$$

– *If* M *is a* $\lambda\mu$ *term of type* A*, then* M^* *is a dual term of type* A*.*

$$\frac{\Gamma \rightharpoonup \Theta \mid M : A}{\Gamma \to \Theta \mid M^* : A}$$

– *If* S *is a* $\lambda\mu$ *statement, then* S^* *is a dual statement.*

$$\frac{\Gamma \mid S \vdash\!\!\rightharpoonup \Theta}{\Gamma \mid S^* \vdash\!\!\rightarrow \Theta}$$

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

Fig. 7. Translation from $\lambda\mu$-calculus to dual calculus

$$
\begin{array}{lll}
(x)_* & \equiv x & (\alpha)_*\{O\} \equiv [\alpha]O \\
(\langle M, N\rangle)_* & \equiv \langle M_*, N_*\rangle & ([K,L])_*\{O\} \equiv L_*\{\mu\beta.\, K_*\{\mu\alpha.\, [\alpha,\beta]O\}\} \\
(\langle M\rangle \mathrm{inl})_* & \equiv \mu[\alpha,\beta].\, [\alpha]M_* & (\mathrm{fst}[K])_*\{O\} \equiv K_*\{\mathrm{fst}\, O\} \\
(\langle N\rangle \mathrm{inr})_* & \equiv \mu[\alpha,\beta].\, [\beta]N_* & (\mathrm{snd}[L])_*\{O\} \equiv L_*\{\mathrm{snd}\, O\} \\
([K]\mathrm{not})_* & \equiv \lambda x.\, K_*\{x\} & (\mathrm{not}\langle M\rangle)_*\{O\} \equiv O\, M_* \\
(\lambda x.N)_* & \equiv \lambda x.\, N_* & (M \mathbin{@} L)_*\{O\} \equiv L_*\{O\, M_*\}
\end{array}
$$

$$((S).\alpha)_* \equiv \mu\alpha.\, S_* \qquad (x.(S))_*\{O\} \equiv (\lambda x.\, S_*)\, O$$

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

Fig. 8. Translation from dual calculus to $\lambda\mu$-calculus

Definition 3. *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.*

$$\frac{\Gamma \rightarrow \Theta \mid M : A}{\Gamma \rightharpoonup \Theta \mid M_* : A}$$

– *If K is a dual coterm of type A, and O is a $\lambda\mu$ term of type A, then $K_*\{O\}$ is a $\lambda\mu$ statement.*

$$\frac{K : A \mid \Gamma \rightarrow \Theta \qquad \Gamma \rightharpoonup \Theta \mid O : A}{\Gamma \mid K_*\{O\} \vdash\!\!\rightarrow \Theta}$$

– *If S is a dual statement, then S_* is a $\lambda\mu$ statement.*

$$\frac{\Gamma \mid S \vdash \Theta}{\Gamma \mid S_* \vdash \Theta}$$

In general, these translations 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 5. *($\lambda\mu$ 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,*

$$\begin{aligned}(M^*)_* &=_v M\\ (S^*)_* &=_v S\end{aligned}$$

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

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

Proposition 6. *(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,*

$$\begin{aligned}(M_*)^* &=^v M\\ (K_*\{O\})^* &=^v O^* \bullet K\end{aligned}$$

$$(S_*)^* \qquad =^v S,$$

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

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

Proposition 7. *($\lambda\mu$ to dual preserves equalities) Translating from the $\lambda\mu$-calculus into the dual calculus preserves call-by-value equalities,*

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

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

The two lines are shown by case analysis on the equations of $\lambda\mu$ that apply to terms and statements respectively.

Proposition 8. *(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, K, L coterms, and S, T statements in dual, and O a term in $\lambda\mu$.

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

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

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

Fig. 9. Duality for the dual calculus

$$
\begin{array}{lll}
(x)_\circ\{O'\} & \equiv [\bar{x}]O' & \\
(\langle M, N\rangle)_\circ\{O'\} & \equiv N_\circ\{\mu\beta.\, M_\circ\{\mu\alpha.\, [\beta, \alpha]O'\}\} & \\
(\mathrm{fst}\, O)_\circ\{O'\} & \equiv (\lambda x.\, O_\circ\{\mu[\beta, \alpha].\, [\alpha]x\})\, O' & ([\alpha]M)_\circ \equiv M_\circ\{\bar{\alpha}\}
\end{array}
$$

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

Fig. 10. Duality for the $\lambda\mu$-calculus

The three lines are shown by case analysis on the equations 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. Since duality is not defined for implication, before applying the results of this section

any occurrences of implication must be translated away, using the results given previously.

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 own inverse – and it carries call-by-value equations into call-by-name equations, 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 4. *The duality translation from the dual calculus to itself is given in Figure 9. It consists of operations on types, terms, coterms, and statements,*

$$A^\circ, \quad M^\circ, \quad K^\circ, \quad S^\circ.$$

- *If A is a type, then A° is the dual type. This extends to environments and coenvironments. If $\Gamma \equiv x_1 : A_1, \ldots, x_m : A_m$, its dual is $\Gamma^\circ \equiv \bar{x}_m : A^\circ_m, \ldots, \bar{x}_1 : A^\circ_1$, and similarly for coenvironments.*
- *If M is a dual term of type A, then M° is a dual coterm of type A.*

$$\frac{}{\Gamma \to \Theta \mid M : A}$$

$$M^\circ : A \mid \Theta^\circ \to \Gamma^\circ$$

– *If K is a dual coterm of type A, and K° is a dual term of type A.*

$$\frac{K : A \mid \Gamma \to \Theta}{\Theta^\circ \to \Gamma^\circ \mid K^\circ : A}$$

– *If S is a dual statement, then S° is a dual statement.*

$$\frac{\Gamma \mid S \vdash\!\!\mapsto \Theta}{\Theta^\circ \mid S^\circ \vdash\!\!\mapsto \Gamma^\circ}$$

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

Proposition 9. *(Involution) Duality is an involution up to identity,*

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

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 con-

firmed by inspection of the reduction rules; indeed, it was the principle guiding their design.

Proposition 10. *(Call-by-value is dual to call-by-name) Duality takes call-by-value equalities into call-by-name equalities, and vice versa.*

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

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

An immediate consequence 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. *The duality transformation from the $\lambda\mu$ calculus to itself is given in Figure 10. It consists of operations on types, terms, and statements, defined as follows,*

$$\begin{array}{ll} A_\circ & \equiv A^\circ \\ M_\circ\{O\} & \equiv ((M^*)^\circ)_*\{O\} \\ S_\circ & \equiv ((S^*)^\circ)_* \end{array}$$

– *If A is a type, then $A_\circ \equiv A^\circ$ is the dual type.*
– *If M is a $\lambda\mu$ term of type A and O is a $\lambda\mu$ term of type $A_\circ$, then $M_\circ\{O\}$ is a $\lambda\mu$ statement.*

$$\frac{\Gamma \rightharpoonup \Theta \mid M : A \qquad \Theta_\circ \rightharpoonup \Gamma_\circ \mid O : A_\circ}{\Theta_\circ \mid M_\circ\{O\} \vdash\!\!\rightharpoonup \Gamma_\circ}$$

– *If S is a $\lambda\mu$ statement, then $S_\circ$ is a $\lambda\mu$ statement.*

$$\frac{\Gamma \mid S \vdash\!\!\rightharpoonup \Theta}{\Theta_\circ \mid S_\circ \vdash\!\!\rightharpoonup \Gamma_\circ}$$

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 11. *(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.*

$$\begin{array}{lll} M =_v N & \textit{iff} & M_\circ\{O\} =_n N_\circ\{O\} \\ S =_v T & \textit{iff} & S_\circ =_n T_\circ \end{array}$$

Here M, N are terms and S, T are 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 12. *(Involution, reloaded) Duality on $\lambda\mu$ is an involution up to equality,*

$$\begin{array}{ll} \mu\alpha.\,(M_\circ\{\bar{\alpha}\})_\circ & =_v M \\ (M_\circ\{O\})_\circ & =_v O_\circ\{M\} \\ (S_\circ)_\circ & =_v S, \end{array}$$

with M, O terms and 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 third line is easy,

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

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

The second line is only slightly harder,

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

The first line follows from the second,

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

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 replaced implication by negation in $\lambda\mu$, yielding a cleaner version of duality. Sometimes less is more!

Acknowledgements

The work on dual calculus owes a great deal to Pierre-Louis Curien, Hugo Herbelin, and Peter Selinger, for their original work and for subsequent discussions. For other comments related to this work, I thank Sebastian Carlier, Olivier Danvy, Roberto Di Cosmo, Marcelo Fiore, Ken-etsu Fujita, Vladimir Gapeyev, Tim Griffin, Daisuke Kimura, Paul Levy, Sam Lindley, William Lovas, Robert McGrail, Rex Page, Bernard Reuss, Ken Shan, Thomas Streicher, Josef Svenningsson, and Steve Zdancewic. Added May 2008: Thanks to Alex Summers for catching a typo in the (name) rule of the dual calculus.

References

1. 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.
2. Vincent Balat, Roberto Di Cosmo and Marcelo Fiore (2004). Extensional normalisation and type-directed partial evaluation for typed lambda-calculus with sums. *Principles of Programming Languages*, 2004.
3. F. Barbanera and S. Berardi (1996). A symmetric lambda calculus for classical program extraction. *Information and Computation*, 125(2):103–117.
4. P. Bernays (1936). Review of "Some Properties of Conversion" by Alonzo Church and J. B. Rosser. *Journal of Symbolic Logic*, 1:74–75.
5. Alonzo Church (1932). A set of postulates for the foundation of logic. *Annals of Mathematics*, II.33:346–366.
6. Alonzo Church (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68.
7. Tristan Crolard (2004). A formulae-as-types interpretation of Subtractive Logic. *Journal of Logic and Computation*, 14(4):529–570.
8. 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.
9. 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.

10. Andrzej Filinski (1989). Declarative continuations and categorical duality. Master's thesis, University of Copenhagen, Copenhagen, Denmark, August 1989. (DIKU Report 89/11.)
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.
12. 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.
13. 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.
14. 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.
15. 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.
16. Eugenio Moggi (1988). Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, Edinburgh University, Department of Computer Science.
17. 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.
18. 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.

19. M. Parigot (1992). $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In LPAR 1992, pages 190–201, Springer-Verlag, LNCS 624.
20. G. D. Plotkin (1975). Call-by-name, call-by-value and the λ-calculus. In *Theoretical Computer Science*, 1:125–159.
21. David Pym and Eike Ritter (2001). On the Semantics of Classical Disjunction. *Journal of Pure and Applied Algebra* 159:315–338.
22. Amr Sabry and Matthias Felleisen (1993). Reasoning about programs in continuation-passing style. *Lisp and Symbolic Computation*, 6(3/4):289–360.
23. Amr Sabry and Philip Wadler (1997). A reflection on call-by-value. In *ACM Transactions on Programming Languages and Systems*, 19(6):916-941.
24. 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.
25. 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.
26. Thomas Streicher and Bernard Reuss (1998). Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.
27. Philip Wadler (2003). Call-by-name is dual to call-by-value. In *International Conference on Functional Programming*, Uppsala, Sweden, 25–29 August 2003.