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

Philip Wadler
Avaya Labs
wadler@avaya.com

ABSTRACT

The rules of classical logic may be formulated in pairs corresponding to De Morgan duals: rules about & are dual to rules about $\vee$. A line of work, including that of Filinski (1989), Griffin (1990), Parigot (1992), Danos, Joinet, and Schellinx (1995), Selinger (1998,2001), and Curien and Herbelin (2000), has led to the startling conclusion that call-by-value is the de Morgan dual of call-by-name.

This paper 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 paper includes crisp formulations of call-by-value and call-by-name that are obviously dual; no similar formulations appear in the literature. The paper gives a CPS translation and its inverse, and shows that the translation is both sound and complete, strengthening a result in Curien and Herbelin (2000).

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, which can be found in the ACM Digital Library archive for ICFP 2003, at

http://portal.acm.org/proceedings/icfp/archive,

or by googling 'wadler dual'.

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

ICFP'03, August 25–29, 2003, Uppsala, Sweden.

1. INTRODUCTION

1.1 Classical logic and duality

Yin and yang, winner and loser, positive and negative, particle and anti-particle, true and false: notions of duality pervade philosophy, science, and mathematics. Theoreticians appreciate duality because it reveals deep symmetries. Practitioners appreciate duality because it offers two-for-the-price-of-one economy.

A salient instance of duality is that between conjunction and disjunction in classical logic, sometimes called de Morgan duality. To find the dual of a proposition, one swaps occurrences of conjunction (&) with occurrences of disjunction ($\vee$) and occurrences of true ($\top$) with occurrences of false ($\bot$), leaving occurrences of negation ($\neg$) unchanged. Two propositions are equivalent if and only if their duals are equivalent.

For example, the Law of Contradiction states that a proposition and its negation are never both true,

$$A \,\&\, \neg A = \bot.$$

Dually, in classical logic, the Law of the Excluded Middle states that either a proposition or its negation is always true,

$$A \vee \neg A = \top.$$

For a second example, & distributes through $\vee$,

$$A \,\&\, (B \vee C) = (A \,\&\, B) \vee (A \,\&\, C).$$

Dually, in classical logic, $\vee$ distributes through &,

$$A \vee (B \,\&\, C) = (A \vee B) \,\&\, (A \vee C).$$

The formulations of logic introduced by Boole (1847) and Frege (1879) do not mention duality. It was first introduced by Schröder (1890), who presented definitions and theorems in pairs, the duals side-by-side in two columns. Schröder was inspired by the duality between points and lines in projective geometry, as introduced by Poncelet (1818) and Gergonne (1826). (See Nidditch (1962) and Grattan-Guinness (2000).)

1.2 Curry-Howard for classical logic

Some of the most important contributions to computing occurred just before the computer was invented. In a series of influential papers, Church (1932,1940) introduced the untyped and typed λ-calculus. And in a single landmark paper, Gentzen (1935) introduced the two formulations of logic most widely used today, natural deduction and sequent calculus, in both intuitionistic and classical variants. (The paper also introduced the use of the symbol $\forall$ for universal quantification.)

Gentzen believed that natural deduction corresponded better to the style of reasoning used in practice (hence the name), but recognized that sequent calculus better reveals the duality of classical logic. Further, Gentzen could demonstrate the consistency of sequent calculus by a method of normalizing proofs called Cut elimination, but he could demonstrate the consistency of natural deduction only by showing its equivalence to sequent calculus.

Only much later did Prawitz (1965) show how to normalize proofs in natural deduction directly. And only later still did Howard (1980) publish a direct correspondence between proofs in intuitionistic natural deduction and terms in typed λ-calculus, with Prawitz's normalization of proofs corresponding to Church's λ-reduction. Similar correspondences between logic and computation were observed by Curry and Feys (1958) and de Bruijn (1968).

What came to be called the Curry-Howard correspondence has proven to be a robust technique for relating a wide range of systems of logic and computation. In particular, Curry-Howard can be applied to classical as well as intuitionistic logic, and to sequent calculus as well as natural deduction.

Sussman and Steele (1975) introduced a call/cc operator in Scheme to capture computing with continuations, and Felleisen *et al.* (1987) introduced the $\mathcal{C}$ operator to model call/cc. (For a fascinating history of continuations see Reynolds (1993).) Griffin (1990) extended the Curry-Howard correspondence to classical logic, by observing that the type of call/cc corresponds to Pierce's Law, and that the type of $\mathcal{C}$ corresponds to the Law of Double Negation.

A refinement of the correspondence between classical logic and computation was given by the $\lambda\mu$-calculus of Parigot (1992,1994). The $\lambda\mu$-calculus corresponds to classical natural deduction in just the same way that λ-calculus corresponds to intuitionistic natural deduction. Call-by-name semantics for the $\lambda\mu$-calculus have been investigated by Ong (1996) and call-by-value semantics by Ong and Stewart (1997).

1.3 Call-by-value and call-by-name

Filinski (1989) was the first to suggest that call-by-value might in some sense be dual to call-by-name in the presence of continuations.

Danos, Joinet, and Schellinx (1995) proposed two dual embeddings of classical logic into linear logic, LKQ and LKT, noting that the first corresponded to call-by-value and the second to call-by-name.

Selinger (1998,2001) modeled the call-by-name semantics of $\lambda\mu$-calculus in a control category, and the call-by-value semantics of $\lambda\mu$-calculus in a dual co-control category.

Curien and Herbelin (2000) further explored this duality using a computational calculus based on sequent calculus, derived from a similar calculus explored earlier by Herbelin (1994). Because sequent calculus displays the dualities of classical logic more clearly than natural deduction, Curien and Parigot's formulation offers some improvements over that of Selinger.

However, in none of these cases is the duality quite as compelling as one might like. Filinski's formulation lacks any connection with logic. Danos, Joinet, and Schellinx's formulation is in terms of proof nets and linear logic, with a less direct connection to computation. Selinger's formulation of duality is not an involution — the dual of the dual of a term is not the original term, but only a term that is equivalent up to isomorphism of types. Curien and Herbelin's formulation is an involution, but to achieve this they must introduce a difference operator as the dual to implication. The computational interpretation of implication $A \supset B$ is a function, but the computational interpretation of the difference $B - A \equiv B \,\&\, \neg A$ is not particularly intuitive.

Barbanera and Berardi (1996) introduce a symmetric λ-calculus, with a clear notion of duality. However, they do not consider either call-by-value or call-by-name reduction, instead their calculus is non-confluent.

1.4 The dual calculus

This paper presents a dual calculus, which corresponds to the classical sequent calculus of Gentzen (1935) in the same way that the λ-calculus of Church (1932,1940) corresponds to the intuitionistic natural deduction of Gentzen (1935).

The approach taken here is to return to the traditional formulation of duality in logic, where conjunction, disjunction, and negation are primitive, and implication is defined in terms of the other connectives. Conjunction ($A \,\&\, B$) corresponds to a product type ($A \times B$), disjunction ($A \vee B$) corresponds to a sum type ($A+B$), and negation ($\neg A$) corresponds to a continuation type ($A \to R$). Implication may be defined in terms of these connectives, though different definitions are required for the call-by-value and call-by-name calculi; one takes $A \supset B \equiv \neg(A \,\&\, \neg B)$ for call-by-value and $A \supset B \equiv \neg A \vee B$ for call-by-name.

The paper includes crisp formulations of call-by-value and call-by-name that are obviously dual; no similar formulations appear in the literature. The paper gives a CPS translation and its inverse, and shows that the translation is both sound and complete.

The paper is organized as follows. Section 2 reviews Gentzen's classical sequent calculus. Section 3 introduces the dual calculus. Section 5 presents call-by-value and call-by-name reduction rules, and observes that they are dual. Section 6 describes call-by-value and call-by-name CPS translations, and shows that they are sound and complete with regard to reductions. Section 7 concludes with a speculation on the dual of call-by-need.

2. GENTZEN'S SEQUENT CALCULUS

Figure 1 presents the syntax and inference rules of the sequent calculus. The rules given here are identical to those in Gentzen (1935), down to the choice of symbols.

Let A, B, C range over formulas, where a formula is either an atomic formula X; a conjunction $A \,\&\, B$; a disjunction $A \vee B$; a negation $\neg A$; or an implication $A \supset B$. Let Γ, Δ range over antecedents and Θ, Λ range over succedents, both of which are sequences of formulas separated by commas. A sequent has the form $\Gamma \rightarrow \Theta$.

The interpretation of a sequent is that the conjunction of the formulas in the antecedent implies the disjunction of the formulas in the succedent. So the sequent

$$A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$$

corresponds to the formula

$$(A_1 \,\&\, \cdots \,\&\, A_m) \supset (B_1 \vee \cdots \vee B_n).$$

A conjunction of zero formulas corresponds to true, and a disjunction of zero formulas corresponds to false.

$$
\begin{array}{lll}
\text{Formula} & A, B, C & ::= \quad X \mid A \,\&\, B \mid A \vee B \mid \neg A \mid A \supset B \\
\text{Antecedent} & \Gamma, \Delta & ::= \quad A_1, \ldots, A_m \\
\text{Succedent} & \Theta, \Lambda & ::= \quad B_1, \ldots, B_n
\end{array}
$$

$$\text{Sequent} \quad \Gamma \rightarrow \Theta$$

$$\frac{}{A \rightarrow A}\,\text{Id}$$

$$\frac{\Gamma \rightarrow \Theta, A \qquad \Gamma \rightarrow \Theta, B}{\Gamma \rightarrow \Theta, A \,\&\, B}\,\&\text{R} \qquad \frac{A, \Gamma \rightarrow \Theta}{A \,\&\, B, \Gamma \rightarrow \Theta} \qquad \frac{B, \Gamma \rightarrow \Theta}{A \,\&\, B, \Gamma \rightarrow \Theta}\,\&\text{L}$$

$$\frac{\Gamma \rightarrow \Theta, A}{\Gamma \rightarrow \Theta, A \vee B} \qquad \frac{\Gamma \rightarrow \Theta, B}{\Gamma \rightarrow \Theta, A \vee B}\,\vee\text{R} \qquad \frac{A, \Gamma \rightarrow \Theta \qquad B, \Gamma \rightarrow \Theta}{A \vee B, \Gamma \rightarrow \Theta}\,\vee\text{L}$$

$$\frac{A, \Gamma \rightarrow \Theta}{\Gamma \rightarrow \Theta, \neg A}\,\neg\text{R} \qquad \frac{\Gamma \rightarrow \Theta, A}{\neg A, \Gamma \rightarrow \Theta}\,\neg\text{L}$$

$$\frac{A, \Gamma \rightarrow \Theta, B}{\Gamma \rightarrow \Theta, A \supset B}\,\supset\text{R} \qquad \frac{\Gamma \rightarrow \Theta, A \qquad B, \Delta \rightarrow \Lambda}{A \supset B, \Gamma, \Delta \rightarrow \Theta, \Lambda}\,\supset\text{L}$$

$$\frac{\Gamma \rightarrow \Theta, A \qquad A, \Delta \rightarrow \Lambda}{\Gamma, \Delta \rightarrow \Theta, \Lambda}\,\text{Cut}$$

$$\frac{\Gamma \rightarrow \Theta}{A, \Gamma \rightarrow \Theta} \quad \text{Thinning} \quad \frac{\Gamma \rightarrow \Theta}{\Gamma \rightarrow \Theta, A}$$

$$\frac{A, A, \Gamma \rightarrow \Theta}{A, \Gamma \rightarrow \Theta} \quad \text{Contraction} \quad \frac{\Gamma \rightarrow \Theta, A, A}{\Gamma \rightarrow \Theta, A}$$

$$\frac{\Delta, A, B, \Gamma \rightarrow \Theta}{\Delta, B, A, \Gamma \rightarrow \Theta} \quad \text{Interchange} \quad \frac{\Gamma \rightarrow \Theta, B, A, \Lambda}{\Gamma \rightarrow \Theta, A, B, \Lambda}$$

Figure 1: Gentzen's sequent calculus

There are *logical* rules for each connective, labeled right or left according as to whether the connective is introduced in the succedent or antecedent. Right rules serve the same purpose as introduction rules in natural deduction, while left rules serve the same purpose as elimination rules.

The remaining rules are *structural*. Id is the obvious axiom, from A one may infer A. Cut combines a proof with A in the succedent and a proof with A in the antecedent to yield a proof with only the other formulas in the antecedents and succedents. Informally, this is justified as follows: from Γ one may infer that either A holds or one of Θ holds; if A holds then from it and Δ one may infer that one of Λ holds; else one of Θ holds. Thinning introduces an additional formula, Contraction replaces two identical formulas by one, and Interchange permutes the order of formulas.

Gentzen proved sequent calculus satisfies a *Cut elimination* property: any proof of a sequent can be transformed to a proof of the same sequent that does not contain Cut. A corrollary of Cut elimination is consistency of the logic: the sequent $\rightarrow$ (which corresponds to true implies false) cannot be the consequence of any rule other than Cut, and is therefore not derivable.

The dual of a formula not containing implication is defined in Figure 2. The dual of conjunction is disjunction and vice versa, and negation is its own dual. The dual of a sequence of formulas is the reverse of the sequence of duals.

PROPOSITION 2.1. *Duality is an involution,*

$$A^{\circ\circ} \quad \equiv \quad A.$$

Rule &R is dual to $\vee$L, &L is dual to $\vee$R, $\neg$R is dual to $\neg$L, Id and Cut are dual to themselves, and Thinning, Contraction, Interchange come in dual pairs. Hence, we have the following.

PROPOSITION 2.2. *A sequent not containing implication is derivable if and only if its dual is derivable.*

$$\Gamma \rightarrow \Theta \quad \textit{iff} \quad \Theta^{\circ} \rightarrow \Gamma^{\circ}.$$

Implication can be defined in terms of other connectives.

PROPOSITION 2.3. *Implication can be defined by*

$$A \supset B \equiv \neg A \vee B \quad \textit{or} \quad A \supset B \equiv \neg(A \,\&\, \neg B).$$

The inference rules for implication can be derived from the inference rules for the other connectives.

$$
\begin{array}{lcl}
(X)^\circ & \equiv & X \\
(A \,\&\, B)^\circ & \equiv & A^\circ \vee B^\circ \\
(A \vee B)^\circ & \equiv & A^\circ \,\&\, B^\circ \\
(\neg A)^\circ & \equiv & \neg A^\circ \\
\\
(A_1, \ldots, A_m)^\circ & \equiv & A_m{}^\circ, \ldots, A_1{}^\circ
\end{array}
$$

Figure 2: Duality for the sequent calculus

3. THE DUAL CALCULUS

The dual calculus is a reformulation of Gentzen's sequent calculus. Under Curry-Howard for natural deduction, terms represent proofs and variables label assumptions. Here terms, coterms, and statements represent proofs, and variables and covariables label antecedents and succedents. Coterms and covariables correspond to what are sometimes called continuation terms and continuation variables.

Figure 3 presents the syntax and inference rules of the dual calculus. The types of the calculus are the same as the formulas of Gentzen's sequent calculus. Let x, y, z range over variables, and α, β, γ range over covariables.

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 $\langle M\rangle$inl or $\langle N\rangle$inr; a complement of a coterm $[K]$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 fst$[K]$ or snd$[L]$; a case $[K, L]$; a complement of a term 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.

There are three kinds of sequents, called right, left, and center, according to whether the proof of the sequent is represented by a term, coterm, or statement. A right sequent has a distinguished formula in the succedent, that is labeled by a term rather than a covariable. A left sequent has a distinguished formula in the antecedent, that is labeled by a coterm rather than a variable. A center sequent has no distinguished formula, and contains a statement.

As with sequent calculus, there are logical rules for each connective. Right rules always end with a right sequent and left rules always end with a left sequent. The & and $\vee$ rules begin and end with the same kind of sequent, while the $\neg$ rules reverse the kind of sequent. For example, &R begins and ends with right sequents, while $\neg$L begins with a right sequent and ends with a left sequent.

The remaining rules are structural rules. Id is split into two forms, one with a right sequent and one with a left sequent. Cut has a right sequent and a left sequent above the line, and ends in a center sequent. There are two new structural rules, RI and LI, that convert a center sequent to an equivalent right or left sequent, by designating a covariable to yield the value of the term, or a variable to consume the value passed to the coterm.

Figure 4 also shows three derived rules. Id is a symmetric form of IdR and IdL that concludes with a center sequent, which cuts a variable against a covariable. RE is an inverse of RI that converts a right sequent into an equivalent center sequent by cutting a term against a covariable. LE is an inverse of LI that converts a left sequent into an equivalent center sequent by cutting a variable against a coterm.

RI and LI behave like introduction rules in natural deduction, and RE and LE behave like elimination rules, hence their names. They also resemble the Activate and Passivate rules in some formulations of the $\lambda\mu$-calculus, such as that presented by Ariola and Herbelin (2003).

Finally, there are eighteen rules for Thinning, Contraction, and Interchange in the antecedent and succedent for right, left, and center sequents. Only the six rules for right sequents are shown, the remaining twelve rules for left and center sequents are similar. The rules shown are not duals; instead, the antecedent rules for right sequents are dual to succedent rules for left sequents, and vice versa, while antecedent and succedent rules for center sequents are dual to each other. In Contraction, substitution of one variable for another is written $M\{x/y\}$, and substitution of one covariable for another is written $K\{\alpha/\beta\}$.

The computational interpretation of a sequent is as follows: one must supply a value for *every* variable (and term) in the antecedent, and the computation will pass a value to *some* continuation variable (or coterm) in the succedent; this corresponds to the fact that the sequent represents the *conjunction* of the formulas in the antecedent and the *disjunction* of the formulas in the succedent. Hence, the computational interpretation of a right sequent

$$x_1 : A_1, \ldots, x_m : A_m \rightarrow \beta_1 : B_1, \ldots, \beta_n : B_n \mid M : B_{n+1}$$

is that if each variable x_i is supplied a value of type A_i then evaluation of the expression M will either return a value of type B_{n+1} *or* pass to some continuation variable β_j a value of type B_j. The computational interpretation of a left sequent

$$K : A_0 \mid x_1 : A_1, \ldots, x_m : A_m \rightarrow \beta_1 : B_1, \ldots, \beta_n : B_n$$

is that if each variable x_i is supplied a value of type A_i *and* a value of type A_0 is supplied to the coterm K, then evaluation will return to some continuation variable β_j a value of type B_j. The computational interpretation of a center sequent

$$x_1 : A_1, \ldots, x_m : A_m \mid S \mapsto \beta_1 : B_1, \ldots, \beta_n : B_n$$

is that if each variable x_i is supplied a value of type A_i then execution of the statement S with pass to some continuation variable β_j a value of type B_j.

The two variable rules correspond to trivial computations. Term x yields the value passed in to variable x. Coterm α consumes a value and passes it out to covariable α.

Computationally, the formula $A \,\&\, B$ corresponds to the product type, where the proof of a conjunction is represented by a pair of the proofs of its subformulas. The term $\langle M, N\rangle$ yields a pair of type $A \,\&\, B$ consisting of the values yielded by terms M of type A and N of type B. The coterm fst$[K]$ consumes a pair of type $A \,\&\, B$, projects out the first component, and passes it on to be consumed by coterm K of type A. Similarly for snd$[L]$.

Dually, the formula $A \vee B$ corresponds to the sum type, where the proof of a disjunction is represented by a proof of

$$
\begin{array}{lll}
\text{Type} & A, B, C & ::= \quad X \mid A \,\&\, B \mid A \vee B \mid \neg A \mid A \supset B \\
\text{Term} & M, N & ::= \quad x \mid \langle M, N \rangle \mid \langle M \rangle \text{inl} \mid \langle N \rangle \text{inr} \mid [K] \text{not} \mid \lambda x.\, N \mid (S).\alpha \\
\text{Coterm} & K, L & ::= \quad \alpha \mid [K, L] \mid \text{fst}[K] \mid \text{snd}[L] \mid \text{not}\langle M \rangle \mid M \mathbin{@} L \mid x.(S) \\
\text{Statement} & S, T & ::= \quad M \bullet K \\
\text{Antecedent} & \Gamma, \Delta & ::= \quad x_1 : A_1, \ldots, x_m : A_m \\
\text{Succedent} & \Theta, \Lambda & ::= \quad \beta_1 : B_1, \ldots, \beta_n : B_n
\end{array}
$$

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

$$
\frac{}{x : A \rightarrowtail \mid x : A}\ \text{IdR} \qquad \frac{}{\alpha : A \mid \rightarrowtail \alpha : A}\ \text{IdL}
$$

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

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

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

$$
\frac{x : A, \Gamma \rightarrowtail \Theta \mid N : B}{\Gamma \rightarrowtail \Theta \mid \lambda x.\, N : A \supset B}\ \supset\text{R} \qquad \frac{\Gamma \rightarrowtail \Theta \mid M : A \qquad L : B \mid \Delta \rightarrowtail \Lambda}{M \mathbin{@} L : A \supset B \mid \Gamma, \Delta \rightarrowtail \Theta, \Lambda}\ \supset\text{L}
$$

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

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A \qquad K : A \mid \Delta \rightarrowtail \Lambda}{\Gamma, \Delta \mid M \bullet K \mapsto \Theta, \Lambda}\ \text{Cut}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : C}{x : A, \Gamma \rightarrowtail \Theta \mid M : C}\ \text{Thinning} \qquad \frac{\Gamma \rightarrowtail \Theta \mid M : C}{\Theta \rightarrowtail \Theta, \alpha : A \mid M : C}
$$

$$
\frac{x : A, y : A, \Gamma \rightarrowtail \Theta \mid M : C}{x : A, \Gamma \rightarrowtail \Theta \mid M\{x/y\} : C}\ \text{Contraction} \qquad \frac{\Gamma \rightarrowtail \Theta, \beta : A, \alpha : A \mid M : C}{\Gamma \rightarrowtail \Theta, \alpha : A \mid M\{\alpha/\beta\} : C}
$$

$$
\frac{\Delta, x : A, y : B, \Gamma \rightarrowtail \Theta \mid M : C}{\Delta, y : B, x : B, \Gamma \rightarrowtail \Theta \mid M : C}\ \text{Interchange} \qquad \frac{\Gamma \rightarrowtail \Theta, \beta : B, \alpha : A, \Lambda \mid M : C}{\Gamma \rightarrowtail \Theta, \alpha : A, \beta : B, \Lambda \mid M : C}
$$

(also Thinning, Contraction, Interchange for center and left sequents)

Figure 3: The dual calculus

$$
\frac{}{x : A \mid x \bullet \alpha \mapsto \alpha : A}\ \text{Id}
$$

$$
\frac{\Gamma \rightarrowtail \Theta \mid M : A}{\Gamma \mid M \bullet \alpha \mapsto \Theta, \alpha : A}\ \text{RE} \qquad \frac{K : A \mid \Gamma \rightarrowtail \Theta}{x : A, \Gamma \mid x \bullet K \mapsto \Theta}\ \text{LE}
$$

Figure 4: Derived structural rules

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

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

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

$$
\begin{array}{lcl}
(x_1 : A_1, \ldots, x_m : A_m)^\circ & \equiv & x_m{}^\circ : A_m{}^\circ, \ldots, x_1{}^\circ : A_1{}^\circ \\
(\beta_1 : B_1, \ldots, \beta_n : B_n)^\circ & \equiv & \beta_n{}^\circ : B_n{}^\circ, \ldots, \beta_1{}^\circ : B_1{}^\circ
\end{array}
$$

Figure 5: Duality for the dual calculus

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \rightarrow \blacksquare\, x : A}\,\mathrm{IdR}}{x : A \rightarrow \blacksquare\, \langle x\rangle\mathrm{inl} : A \vee \neg A}\,\vee\mathrm{R}}{x : A \blacksquare\, \langle x\rangle\mathrm{inl} \bullet \gamma \mapsto \gamma : A \vee \neg A}\,\mathrm{RE}}{x.(\langle x\rangle\mathrm{inl} \bullet \gamma) : A \blacksquare\, \rightarrow \gamma : A \vee \neg A}\,\mathrm{LI}}{\rightarrow \gamma : A \vee \neg A \blacksquare\, [x.(\langle x\rangle\mathrm{inl} \bullet \gamma)]\mathrm{not} : \neg A}\,\neg\mathrm{R}}{\rightarrow \gamma : A \vee \neg A \blacksquare\, \langle [x.(\langle x\rangle\mathrm{inl} \bullet \gamma)]\mathrm{not}\rangle\mathrm{inr} : A \vee \neg A}\,\vee\mathrm{R}}{\blacksquare\, \langle [x.(\langle x\rangle\mathrm{inl} \bullet \gamma)]\mathrm{not}\rangle\mathrm{inr} \bullet \delta \mapsto \gamma : A \vee \neg A, \delta : A \vee \neg A}\,\mathrm{RE}}{\blacksquare\, \langle [x.(\langle x\rangle\mathrm{inl} \bullet \gamma)]\mathrm{not}\rangle\mathrm{inr} \bullet \gamma \mapsto \gamma : A \vee \neg A}\,\mathrm{Cont}}{\rightarrow \blacksquare\, (\langle [x.(\langle x\rangle\mathrm{inl} \bullet \gamma)]\mathrm{not}\rangle\mathrm{inr} \bullet \gamma).\gamma : A \vee \neg A}\,\mathrm{RI}
$$

Figure 6: Law of the excluded middle

either its left or right subformula. The term $\langle M\rangle\mathrm{inl}$ yields a value of type $A \vee B$ consisting of the injection on the left of the value yielded by term M of type A. Similarly for $\langle N\rangle\mathrm{inr}$. The coterm $[K, L]$ is like a case expression: it consumes a value of type $A \vee B$, and depending on whether it is a left or right injection, passes on the injected value to be consumed by coterm K of type A or coterm L of type B.

Logically, the formula $\neg A$ is equivalent to the implication $A \supset \bot$. This suggests that computationally the formula $\neg A$ should correspond to a continuation, where a continuation is a function that accepts a value and returns nothing. (One may recall the song about Charlie on the MTA: "And did he ever return, no he never returned, and his fate is still unlearned.") Though it never returns, a continuation may still yield a value through one of its free covariables, just as it may consume values other than its argument through its free variables. The term $[K]\mathrm{not}$ yields a continuation of type $\neg A$ that accepts an argument of type A and passes it to be consumed by the coterm K. The term $\mathrm{not}\langle M\rangle$ consumes a continuation of type $\neg A$ by passing it as argument the value of the term M of type A.

The formula $A \supset B$ corresponds to a function type. The term $\lambda x.\, N$ yields a function of type $A \supset B$ that takes an argument x of type A and yields the value of term N of type B. The coterm $M \mathbin{@} L$ consumes a function of type $A \supset B$ by passing it as argument the value of the term M of type A, and then passing the result to be consumed by the coterm L of type B.

Cut plugs together a term and a coterm. The statement $M \bullet K$ takes the value yielded by term M and passes it to be consumed by coterm K.

In RI, the term $(S).\alpha$ executes statement S and yields the value of type A passed to the covariable α. In LI, the coterm $x.(S)$ consumes a value of type A which is bound to the variable x and then executes statement S.

Rules Cut, Id, RE, and LE overlap; we can avoid the overlap by using Cut only when the term is not a variable and the coterm is not a covariable; using RE only when the term is not a variable, and using LE only when the coterm is not a covariable. Then every term, coterm, and statement still has a unique proof tree (up to structural rules), and the correspondence with sequent calculus is more closely preserved. A proof in the sequent calculus is mapped into the dual calculus by adding occurrences of RI, LI, RE, and LE as needed; and a proof in the dual calculus is mapped into the sequent calculus by eliding the occurrences of RI, LI, RE, and LE. (Avoiding overlap in this way resolves the problem mentioned by Curien and Herbelin (2000) in the paragraph spanning pages 234–5 that ends "no perfect world".)

Types, terms, coterms, statements not involving implication have a dual, as defined in Figure 5. The definition assumes a bijection mapping each variable x into a covariable x°, with its inverse mapping each covariable α into a variable α°, such that $x^{\circ\circ} \equiv x$ and $\alpha^{\circ\circ} \equiv \alpha$.

PROPOSITION 3.1. *Duality is an involution,*

$$
\begin{array}{lcl}
A^{\circ\circ} & \equiv & A \\
M^{\circ\circ} & \equiv & M \\
K^{\circ\circ} & \equiv & K \\
S^{\circ\circ} & \equiv & S.
\end{array}
$$

PROPOSITION 3.2. *A sequent not containing implication is derivable if and only if its dual is derivable,*

$$
\left.\begin{array}{r}
\Gamma \rightarrow \Theta \blacksquare\, M : A \\
K : A \blacksquare\, \Gamma \rightarrow \Theta \\
\Gamma \blacksquare\, S \mapsto \Theta
\end{array}\right\} \textit{ iff } \left\{\begin{array}{l}
M^\circ : A^\circ \blacksquare\, \Theta^\circ \rightarrow \Gamma^\circ \\
\Theta^\circ \rightarrow \Gamma^\circ \blacksquare\, K^\circ : A^\circ \\
\Theta^\circ \blacksquare\, S^\circ \mapsto \Gamma^\circ.
\end{array}\right.
$$

As with sequent calculus, implication can be defined in terms of the other connectives. We will return to this point after considering an extended example, the law of the excluded middle, and considering the reduction rules for call-by-value and call-by-name.

4. EXAMPLE: EXCLUDED MIDDLE

Figure 6 shows a proof of the law of the excluded middle, $A \vee \neg A$. The computational interpretation of this proof exploits the ability of classisal control operators to return multiple times from a single term. The term of type $A \vee \neg A$ first returns an injection into the right of the sum, a continuation that expects a value of type A. If the continuation is ever passed such a value, then the original term of type $A \vee \neg A$ now returns an injection into the left of the sum, containing the value passed to the continuation.

The following story illustrates this behavior. (With apologies to Peter Selinger, who tells a similar story about a king, a wizard, and the Philosopher's stone.)

Once upon a time, the devil approached a man and made an offer: "Either (a) I will give you one billion dollars, or (b) I will grant you any wish if you pay me one billion dollars. Of course, I get to choose whether I offer (a) or (b)."

The man was wary. Did he need to sign over his soul? No, said the devil, all the man need do is accept the offer.

The man pondered. If he was offered (b) it was unlikely that he would ever be able to buy the wish, but what was the harm in having the opportunity available?

"I accept," said the man at last. "Do I get (a) or (b)?"

The devil paused. "I choose (b)."

The man was disappointed but not surprised. That was that, he thought. But the offer gnawed at him. Imagine what he could do with his wish! Many years passed, and the man began to accumulate money. To get the money he sometimes did bad things, and dimly he realized that this must be what the devil had in mind. Eventually he had his billion dollars, and the devil appeared again.

"Here is a billion dollars," said the man, handing over a valise containing the money. "Grant me my wish!"

The devil took possession of the valise. Then he said, "Oh, did I say (b) before? I'm so sorry. I meant (a). It is my great pleasure to give you one billion dollars."

And the devil handed back to the man the same valise that the man had just handed to him.

5. REDUCTIONS

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.

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

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.

$$\begin{array}{ccc} & (x \bullet \alpha).\alpha \bullet y.(y \bullet \beta) & \\ \swarrow & & \searrow \\ x \bullet y.(y \bullet \beta) & & (x \bullet \alpha).\alpha \bullet \beta \\ \searrow & & \swarrow \\ & x \bullet \beta & \end{array}$$

But sometimes they are not.

$$\begin{array}{ccc} & (x \bullet \alpha).\beta \bullet y.(z \bullet \gamma) & \\ \swarrow & & \searrow \\ x \bullet \alpha & & z \bullet \gamma \end{array}$$

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.

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

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.

$$\begin{array}{lll} (\beta\mathrm{L}) & M \bullet x.(S) & \longrightarrow_n \quad S\{M/x\} \\ (\beta\mathrm{R}) & (S).\alpha \bullet P & \longrightarrow_n \quad S\{P/\alpha\} \end{array}$$

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 7. 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.

A context is a term or coterm that contains a hole which may be filled with a term. Let E range over term contexts. The hole is written $\{\,\}$, and the result of filling the hole in a term context E with a term M is written $E\{M\}$.

Reductions ($\beta\&$), ($\beta\vee$), ($\beta\neg$), and ($\beta\supset$) are *logical* reductions, and correspond to cutting a right rule against a left rule. For instance, the ($\beta\&$) reductions correspond to the following familiar reductions from lambda calculus.

$$\begin{array}{lll} (\beta\&) & \mathrm{fst}\ \langle V, W\rangle & \longrightarrow_v \quad V \\ (\beta\&) & \mathrm{snd}\ \langle V, W\rangle & \longrightarrow_v \quad W \end{array}$$

The remaining reductions are *structural*. Reductions (βL) and (βR) correspond to following an introduction rule by an elimination rule. Reductions (ηL) and (ηR) correspond to following an elimination rule by an introduction rule. Reduction (ς) is a commuting rule.

Reductions (ηL), (ηR), and (ς) are in fact expansions. To avoid infinite regress, expansions (ηL) and (ηR) should be applied only to a term M or coterm K that is not the immediate subject of a cut, and expansion (ς) should be applied only when the term M is not a value.

Unlike (βL), there is no need to restrict to values in (ηL).

Reduction (ς) introduces new bindings for every subterm that is not a variable. It is similar to the reductions (let.1) and (let.2) in the λ_c-calculus of Moggi (1988).

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

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

It is claimed without proof that the reductions are confluent. It is also claimed that if reductions (ηL), (ηR), and (ς) are omitted, then the remaining reductions are strongly normalizing for typed terms.

Call-by-name reductions are shown in Figure 8. With the exception of implication, they are dual to call-by-value.

PROPOSITION 5.1. *For terms, coterms, and statements not involving implication, call-by-value is dual to call-by-name,*

$$\left.\begin{array}{l} M \longrightarrow_v N \\ K \longrightarrow_v L \\ S \longrightarrow_v T \end{array}\right\} \textit{ iff } \left\{\begin{array}{l} M^\circ \longrightarrow_n N^\circ \\ K^\circ \longrightarrow_n L^\circ \\ S^\circ \longrightarrow_n T^\circ. \end{array}\right.$$

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

$(\beta\&)$	$\langle V, W\rangle \bullet \mathrm{fst}[K]$	$\longrightarrow_v$	$V \bullet K$	
$(\beta\&)$	$\langle V, W\rangle \bullet \mathrm{snd}[L]$	$\longrightarrow_v$	$W \bullet L$	
$(\beta\vee)$	$\langle V\rangle\mathrm{inl} \bullet [K, L]$	$\longrightarrow_v$	$V \bullet K$	
$(\beta\vee)$	$\langle W\rangle\mathrm{inr} \bullet [K, L]$	$\longrightarrow_v$	$W \bullet L$	
$(\beta\neg)$	$[K]\mathrm{not} \bullet \mathrm{not}\langle M\rangle$	$\longrightarrow_v$	$M \bullet K$	
$(\beta\supset)$	$\lambda x.\, N \bullet V \mathbin{@} L$	$\longrightarrow_v$	$V \bullet x.(N \bullet L)$	
$(\beta\mathrm{L})$	$V \bullet x.(S)$	$\longrightarrow_v$	$S\{V/x\}$	
$(\beta\mathrm{R})$	$(S).\alpha \bullet K$	$\longrightarrow_v$	$S\{K/\alpha\}$	
$(\eta\mathrm{L})$	K	$\longrightarrow_v$	$x.(x \bullet K)$	if $x \notin \mathrm{free}(K)$
$(\eta\mathrm{R})$	M	$\longrightarrow_v$	$(M \bullet \alpha).\alpha$	if $\alpha \notin \mathrm{free}(M)$
(ς)	$E\{M\}$	$\longrightarrow_v$	$(M \bullet x.(E\{x\} \bullet \beta)).\beta$	

Figure 7: Call-by-value reductions

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

$(\beta\&)$	$\langle M, N\rangle \bullet \mathrm{fst}[P]$	$\longrightarrow_n$	$M \bullet P$	
$(\beta\&)$	$\langle M, N\rangle \bullet \mathrm{snd}[Q]$	$\longrightarrow_n$	$N \bullet Q$	
$(\beta\vee)$	$\langle M\rangle\mathrm{inl} \bullet [P, Q]$	$\longrightarrow_n$	$M \bullet P$	
$(\beta\vee)$	$\langle N\rangle\mathrm{inr} \bullet [P, Q]$	$\longrightarrow_n$	$N \bullet Q$	
$(\beta\neg)$	$[K]\mathrm{not} \bullet \mathrm{not}\langle M\rangle$	$\longrightarrow_n$	$M \bullet K$	
$(\beta\supset)$	$\lambda x.\, N \bullet M \mathbin{@} Q$	$\longrightarrow_n$	$M \bullet x.(N \bullet Q)$	
$(\beta\mathrm{L})$	$M \bullet x.(S)$	$\longrightarrow_n$	$S\{M/x\}$	
$(\beta\mathrm{R})$	$(S).\alpha \bullet P$	$\longrightarrow_n$	$S\{P/\alpha\}$	
$(\eta\mathrm{L})$	K	$\longrightarrow_n$	$x.(x \bullet K)$	if $x \notin \mathrm{free}(K)$
$(\eta\mathrm{R})$	M	$\longrightarrow_n$	$(M \bullet \alpha).\alpha$	if $\alpha \notin \mathrm{free}(M)$
(ς)	$F\{K\}$	$\longrightarrow_n$	$y.((y \bullet F\{\alpha\}).\alpha \bullet K)$	

Figure 8: Call-by-name reductions

Implication can be defined in terms of the other operators. Under call-by-value function abstractions must translate to values, while under call-by-name function applications must translate to covalues, and this forces different definitions for the two reduction disciplines.

PROPOSITION 5.2. *Under call-by-value, implication can be defined by*

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

The translation of a function abstraction is a value, and the inference and reduction rules for implication can be derived from the inference rules for the other connectives.

PROPOSITION 5.3. *Under call-by-name, implication can be defined by*

$$\begin{array}{lcl} A \supset B & \equiv & \neg A \vee B \\ \lambda x.\, N & \equiv & (\langle [x.(\langle N\rangle \mathit{inr} \bullet \gamma)]\mathit{not}\rangle \mathit{inl} \bullet \gamma).\gamma \\ M \mathbin{@} L & \equiv & [\mathit{not}\langle M\rangle, L]. \end{array}$$

The translation of a function application is a covalue, and the inference and reduction rules for implication can be derived from the inference rules for the other connectives.

6. CONTINUATION-PASSING STYLE

Plotkin (1975) formalized the call-by-value λ_v-calculus and its corresponding continuation-passing style (CPS) translation. He showed that the CPS translation is *sound* in that it preserves reductions, but not *complete* in that it does not reflect equalities. Sabry and Felleisen (1993) sharpened Plotkin's result by taking as their source the λ_c-calculus of Moggi (1988). They showed that the CPS translation from λ_c is both sound and complete, in that it is an *equational correspondence* that both preserves and reflects equalities. Sabry and Wadler (1997) sharpened this result further. They showed that the CPS translation is a *Galois connection* that both preserves and reflects reductions. Here we give an analogous result for the CPS translation from the dual calculus, strengthening a result of Curien and Herbelin (2000).

The call-by-value CPS translation is shown in Figure 9, and the call-by-name CPS translation is shown in Figure 10. We write $(A)^V, (V)^V, (M)^v, (K)^v, (S)^v$ for the call-by-value translation of types, values, terms, coterms, and statements, and similarly for call-by-name.

The call-by-value CPS translation resembles that for the λ_v-calculus in Plotkin (1975), and the call-by-name CPS translation resembles that for the $\lambda\mu$-calculus in Hofmann

$$
\begin{array}{lcl}
(X)^V & \equiv & X \\
(A \,\&\, B)^V & \equiv & (A)^V \times (B)^V \\
(A \vee B)^V & \equiv & (A)^V + (B)^V \\
(\neg A)^V & \equiv & (A)^V \to R
\end{array}
\qquad
\begin{array}{lcl}
(x)^V & \equiv & x \\
(\langle V, W \rangle)^V & \equiv & \langle (V)^V, (W)^V \rangle \\
(\langle V \rangle \mathrm{inl})^V & \equiv & \mathrm{inl}\ (V)^V \\
(\langle W \rangle \mathrm{inr})^V & \equiv & \mathrm{inr}\ (W)^V \\
([K]\mathrm{not})^V & \equiv & (K)^v
\end{array}
$$

$$
\begin{array}{lcl}
(x)^v & \equiv & \boldsymbol{\lambda} \gamma.\, \gamma\, x \\
(\langle M, N \rangle)^v & \equiv & \boldsymbol{\lambda} \gamma.\, (M)^v\, (\boldsymbol{\lambda} x.\, (N)^v\, (\boldsymbol{\lambda} y.\, \gamma\, \langle x, y \rangle)) \\
(\langle M \rangle \mathrm{inl})^v & \equiv & \boldsymbol{\lambda} \gamma.\, (M)^v\, (\boldsymbol{\lambda} x.\, \gamma\, (\mathrm{inl}\ x)) \\
(\langle N \rangle \mathrm{inr})^v & \equiv & \boldsymbol{\lambda} \gamma.\, (N)^v\, (\boldsymbol{\lambda} y.\, \gamma\, (\mathrm{inr}\ y)) \\
([K]\mathrm{not})^v & \equiv & \boldsymbol{\lambda} \gamma.\, \gamma\, (\boldsymbol{\lambda} z.\, (K)^v\, z) \\
((S).\alpha)^v & \equiv & \boldsymbol{\lambda} \alpha.\, (S)^v
\end{array}
\qquad
\begin{array}{lcl}
(\alpha)^v & \equiv & \boldsymbol{\lambda} z.\, \alpha\, z \\
([K, L])^v & \equiv & \boldsymbol{\lambda} z.\, \mathrm{case}\ z\ \mathrm{of\ inl}\ x \Rightarrow (K)^v\, x,\ \mathrm{inr}\ y \Rightarrow (L)^v\, y \\
(\mathrm{fst}[K])^v & \equiv & \boldsymbol{\lambda} z.\, \mathrm{case}\ z\ \mathrm{of}\ \langle x, - \rangle \Rightarrow (K)^v\, x \\
(\mathrm{snd}[L])^v & \equiv & \boldsymbol{\lambda} z.\, \mathrm{case}\ z\ \mathrm{of}\ \langle -, y \rangle \Rightarrow (L)^v\, y \\
(\mathrm{not}\langle M \rangle)^v & \equiv & \boldsymbol{\lambda} z.\, (\boldsymbol{\lambda} \gamma.\, (M)^v\, \gamma)\, z \\
(x.(S))^v & \equiv & \boldsymbol{\lambda} x.\, (S)^v
\end{array}
$$

$$
(M \bullet K)^v \quad \equiv \quad (M)^v\, (K)^v
$$

Figure 9: Call-by-value CPS translation

$$
\begin{array}{lcl}
(X)^N & \equiv & X \\
(A \,\&\, B)^N & \equiv & (A)^N + (B)^N \\
(A \vee B)^N & \equiv & (A)^N \times (B)^N \\
(\neg A)^N & \equiv & (A)^N \to R
\end{array}
\qquad
\begin{array}{lcl}
(\alpha)^N & \equiv & \alpha \\
([P, Q])^N & \equiv & \langle (P)^N, (Q)^N \rangle \\
(\mathrm{fst}[P])^N & \equiv & \mathrm{inl}\ (P)^N \\
(\mathrm{snd}[Q])^N & \equiv & \mathrm{inr}\ (Q)^N \\
(\mathrm{not}\langle M \rangle)^N & \equiv & (M)^n
\end{array}
$$

$$
\begin{array}{lcl}
(x)^n & \equiv & \boldsymbol{\lambda} \gamma.\, x\, \gamma \\
(\langle M, N \rangle)^n & \equiv & \boldsymbol{\lambda} \gamma.\, \mathrm{case}\ \gamma\ \mathrm{of\ inl}\ \alpha \Rightarrow (M)^n\, \alpha,\ \mathrm{inr}\ \beta \Rightarrow (N)^n\, \beta \\
(\langle M \rangle \mathrm{inl})^n & \equiv & \boldsymbol{\lambda} \gamma.\, \mathrm{case}\ \gamma\ \mathrm{of}\ \langle \alpha, - \rangle \Rightarrow (M)^n\, \alpha \\
(\langle N \rangle \mathrm{inr})^n & \equiv & \boldsymbol{\lambda} \gamma.\, \mathrm{case}\ \gamma\ \mathrm{of}\ \langle -, \beta \rangle \Rightarrow (N)^n\, \beta \\
([K]\mathrm{not})^n & \equiv & \boldsymbol{\lambda} \gamma.\, (\boldsymbol{\lambda} z.\, (K)^n\, z)\, \gamma \\
((S).\alpha)^n & \equiv & \boldsymbol{\lambda} \alpha.\, (S)^n
\end{array}
\qquad
\begin{array}{lcl}
(\alpha)^n & \equiv & \boldsymbol{\lambda} z.\, z\, \alpha \\
([K, L])^n & \equiv & \boldsymbol{\lambda} z.\, (K)^n\, (\boldsymbol{\lambda} \alpha.\, (L)^n\, (\boldsymbol{\lambda} \beta.\, z\, \langle \alpha, \beta \rangle)) \\
(\mathrm{fst}[K])^n & \equiv & \boldsymbol{\lambda} z.\, (K)^n\, (\boldsymbol{\lambda} \alpha.\, z\, (\mathrm{inl}\ \alpha)) \\
(\mathrm{snd}[L])^n & \equiv & \boldsymbol{\lambda} z.\, (L)^n\, (\boldsymbol{\lambda} \beta.\, z\, (\mathrm{inr}\ \beta)) \\
(\mathrm{not}\langle M \rangle)^n & \equiv & \boldsymbol{\lambda} z.\, z\, (\boldsymbol{\lambda} \gamma.\, (M)^n\, \gamma) \\
(x.(S))^n & \equiv & \boldsymbol{\lambda} x.\, (S)^n
\end{array}
$$

$$
(M \bullet K)^n \quad \equiv \quad (K)^n\, (M)^n
$$

Figure 10: Call-by-name CPS translation

and Streicher (1997). Similar translations are presented by Curien and Herbelin (2000) and Selinger (2001).

The source of the translations is the dual calculus without implications, and the target is a restriction of the simply typed λ-calculus. The syntax and reductions of the target are shown in Figure 11. The typing rules for the target are not shown, as they are standard. The target calculus is *indifferent*, in the sense of Plotkin (1975), in that the arguments of all reductions are values, so the reductions are equally valid under both call-by-value and call-by-name.

PROPOSITION 6.1. *The call-by-value CPS translation preserves types.*

$$
\left.\begin{array}{r} \Gamma \to \Theta \mid V : A \\ \Gamma \to \Theta \mid M : A \\ K : A \mid \Gamma \to \Theta \\ \Gamma \mid S \mapsto \Theta \end{array}\right\} \textit{ iff } \left\{\begin{array}{l} (\Gamma)^V, (\neg\Theta)^V \to (V)^V : (A)^V \\ (\Gamma)^V, (\neg\Theta)^V \to (M)^v : (\neg\neg A)^V \\ (\Gamma)^V, (\neg\Theta)^V \to (K)^v : (\neg A)^V \\ (\Gamma)^V, (\neg\Theta)^V \to (S)^v : R \end{array}\right.
$$

PROPOSITION 6.2. *The call-by-name CPS translation preserves types.*

$$
\left.\begin{array}{r} P : A \mid \Gamma \to \Theta \\ \Gamma \to \Theta \mid M : A \\ K : A \mid \Gamma \to \Theta \\ \Gamma \mid S \mapsto \Theta \end{array}\right\} \textit{ iff } \left\{\begin{array}{l} (\neg\Gamma)^N, (\Theta)^N \to (P)^N : (A)^N \\ (\neg\Gamma)^N, (\Theta)^N \to (M)^n : (\neg A)^N \\ (\neg\Gamma)^N, (\Theta)^N \to (K)^n : (\neg\neg A)^N \\ (\neg\Gamma)^N, (\Theta)^N \to (S)^n : R \end{array}\right.
$$

PROPOSITION 6.3. *The call-by-value and call-by-name CPS translations are dual.*

$$
\begin{array}{lcl}
(A)^V & \equiv & (A^\circ)^N \\
(V)^V & \equiv & (V^\circ)^N \\
(M)^v & \equiv & (M^\circ)^n \\
(K)^v & \equiv & (K^\circ)^n \\
(S)^v & \equiv & (S^\circ)^n
\end{array}
$$

There is an intuitive explanation of the duality between the CPS translations, derived directly from de Morgan's laws relating conjunction and disjunction. Consider the continuation of a term of pair type $A \,\&\, B$. Depending on whether the term is deconstructed using fst[] or snd[], either the continuation will select the first component of type A or the second component of type B, and then continue evaluation. Thus, the continuation for a pair $\neg(A \,\&\, B)$ is a sum of continuations $\neg A \vee \neg B$. Now consider the continuation of a term of sum type $A \vee B$. Since the term may be constructed using either $\langle\rangle$inl or $\langle\rangle$inr, the continuation must be prepared to accept both a left injection of type A and a right injection of type B. Thus, the continuation for a sum $\neg(A \vee B)$ is a pair of continuations $\neg A \,\&\, \neg B$.

The CPS translations preserve and reflect reductions. We consider only the call-by-value case, as the call-by-name case

$$
\begin{array}{lrcl}
\text{Type} & A, B & ::= & X \mid A \times B \mid A + B \mid A \to R \\
\\
\text{Value} & V, W & ::= & x \mid \langle V, W \rangle \mid \text{inl}\ V \mid \text{inr}\ W \mid K \\
\text{Term} & M, N & ::= & \lambda\alpha.\, S \\
\text{Coterm} & K, L & ::= & \lambda x.\, S \\
\text{Statement} & S, T & ::= & \alpha\, V \mid \\
 & & & \text{case}\ V\ \text{of}\ \langle x, - \rangle \Rightarrow S \mid \\
 & & & \text{case}\ V\ \text{of}\ \langle -, y \rangle \Rightarrow T \mid \\
 & & & \text{case}\ V\ \text{of inl}\ x \Rightarrow S, \text{inr}\ y \Rightarrow T \mid \\
 & & & M\, V
\end{array}
$$

$$
\begin{array}{lll}
(\beta\times) & \text{case}\ \langle V, W \rangle\ \text{of}\ \langle x, - \rangle \Rightarrow S & \longrightarrow\ S\{V/x\} \\
(\beta\times) & \text{case}\ \langle V, W \rangle\ \text{of}\ \langle -, y \rangle \Rightarrow T & \longrightarrow\ T\{W/y\} \\
(\beta+) & \text{case inl}\ V\ \text{of inl}\ x \Rightarrow S, \text{inr}\ y \Rightarrow T & \longrightarrow\ S\{V/x\} \\
(\beta+) & \text{case inr}\ W\ \text{of inl}\ x \Rightarrow S, \text{inr}\ y \Rightarrow T & \longrightarrow\ T\{W/y\} \\
(\beta\to) & (\lambda\alpha.\, S)\,(\lambda x.\, T) & \longrightarrow\ S\{T\{-/x\}/\alpha\,-\}
\end{array}
$$

Figure 11: CPS target calculus

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

Figure 12: Kernel of the call-by-value dual calculus

$$
\begin{array}{lcl@{\qquad}lcl}
(X)_V & \equiv & X & (x)_V & \equiv & x \\
(A \times B)_V & \equiv & (A)_V \,\&\, (B)_V & (\langle V, W \rangle)_V & \equiv & \langle (V)_V, (W)_V \rangle \\
(A + B)_V & \equiv & (A)_V \vee (B)_V & (\text{inl}\ V)_V & \equiv & \langle (V)_V \rangle\text{inl} \\
(A \to R)_V & \equiv & \neg(A)_V & (\text{inr}\ W)_V & \equiv & \langle (W)_V \rangle\text{inr} \\
 & & & (K)_V & \equiv & [(K)_v]\text{not}
\end{array}
$$

$$
(\lambda\alpha.\, S)_v \equiv ((S)_v).\alpha \qquad (\lambda x.\, S)_v \equiv x.((S)_v)
$$

$$
\begin{array}{lcl}
(\alpha\, V)_v & \equiv & (V)_V \bullet \alpha \\
(\text{case}\ V\ \text{of}\ \langle x, - \rangle \Rightarrow S)_v & \equiv & (V)_V \bullet \text{fst}[x.((S)_v)] \\
(\text{case}\ V\ \text{of}\ \langle -, y \rangle \Rightarrow T)_v & \equiv & (V)_V \bullet \text{snd}[y.((T)_v)] \\
(\text{case}\ V\ \text{of inl}\ x \Rightarrow S, \text{inr}\ y \Rightarrow T)_v & \equiv & (V)_V \bullet [x.((S)_v), y.((T)_v)] \\
(M\, V)_v & \equiv & (V)_V \bullet \text{not}\langle (M)_v \rangle
\end{array}
$$

Figure 13: Inverse call-by-value CPS translation

is entirely dual. The proof given here is based on that of Sabry and Wadler (1997).

In the CPS translations, λ-abstractions in boldface are *administrative*, that is, they are reduced at the time the translation is performed. Figure 14 presents five examples, showing the translation before and after performing the administrative reductions.

The target syntax in Figure 11 is closed with respect to substitutions. The notation $S\{V/x\}$ stands for statement S with all occurrences of variable x replaced by value V, and is again a valid statement in the target. Note it may not be valid to substitute a term that is not a value for a variable. The notation $S\{T\{-/x\}/\alpha\,-\}$ stands for statement S with all occurrences of the form $\alpha\, V$ replaced by $T\{V/x\}$. Note that all occurrences of α in a valid statement in the target have the form $\alpha\, V$ for some value V, and the result is again a legal statement in the target. The target is also closed with respect to reductions, in that reduction of a term in the target syntax always yields a term in the target syntax.

The key intuition behind the proofs is to note that there is a kernel of the dual calculus that is in one-to-one correspondence with the CPS target calculus. This kernel is shown in

$$
\begin{array}{ll}
 & (\langle x, y\rangle)^v \\
\equiv & \boldsymbol{\lambda}\gamma.\,(\boldsymbol{\lambda}\gamma.\,\gamma\,x)\,(\boldsymbol{\lambda}x'.\,(\boldsymbol{\lambda}\gamma.\,\gamma\,y)\,(\boldsymbol{\lambda}y'.\,\gamma\,\langle x', y'\rangle)) \\
\equiv & \lambda\gamma.\,\gamma\,\langle x, y\rangle
\end{array}
$$

$$
\begin{array}{ll}
 & ((z \bullet \mathrm{fst}[\alpha]).\alpha)^v \\
\equiv & \boldsymbol{\lambda}\alpha.\,(\boldsymbol{\lambda}\gamma.\,\gamma\,z)\,(\boldsymbol{\lambda}z'.\,\mathrm{case}\ z'\ \mathrm{of}\ \langle x, -\rangle \Rightarrow (\boldsymbol{\lambda}z''.\,\alpha\,z'')\,x) \\
\equiv & \lambda\alpha.\,\mathrm{case}\ z\ \mathrm{of}\ \langle x, -\rangle \Rightarrow \alpha\,x
\end{array}
$$

$$
\begin{array}{ll}
 & (\langle (z \bullet \mathrm{fst}[\alpha]).\alpha, (z \bullet \mathrm{snd}[\beta]).\beta\rangle)^v \\
\equiv & \boldsymbol{\lambda}\gamma.\,(\boldsymbol{\lambda}\alpha.\,\mathrm{case}\ z\ \mathrm{of}\ \langle x, -\rangle \Rightarrow \alpha\,x) \\
 & \qquad (\boldsymbol{\lambda}x'.\,(\boldsymbol{\lambda}\beta.\,\mathrm{case}\ z\ \mathrm{of}\ \langle -, y\rangle \Rightarrow \beta\,y) \\
 & \qquad\quad (\boldsymbol{\lambda}y'.\,\gamma\,\langle x', y'\rangle)) \\
\equiv & \lambda\gamma.\,\mathrm{case}\ z\ \mathrm{of}\ \langle x, -\rangle \Rightarrow \mathrm{case}\ z\ \mathrm{of}\ \langle -, y\rangle \Rightarrow \gamma\,\langle x, y\rangle
\end{array}
$$

$$
\begin{array}{ll}
 & (x \bullet \alpha)^v \\
\equiv & (\boldsymbol{\lambda}\gamma.\,\gamma\,x)\,(\boldsymbol{\lambda}z.\,\alpha\,z) \\
\equiv & \alpha\,x
\end{array}
$$

$$
\begin{array}{ll}
 & ([\alpha]\mathrm{not} \bullet \mathrm{not}\langle x\rangle)^v \\
\equiv & (\boldsymbol{\lambda}\gamma.\,\gamma\,(\lambda z.\,(\boldsymbol{\lambda}z.\,\alpha\,z)z))\,(\boldsymbol{\lambda}z.\,(\lambda\gamma.\,(\boldsymbol{\lambda}\gamma.\,x\,\gamma)\,\gamma)\,z) \\
\equiv & (\lambda\gamma.\,x\,\gamma)\,(\lambda z.\,\alpha\,z)
\end{array}
$$

Figure 14: Examples of CPS translation

$$
\begin{array}{ll}
 & ((\langle x, y\rangle)^v)_v \\
\equiv & \langle x, y\rangle
\end{array}
$$

$$
\begin{array}{ll}
 & (((z \bullet \mathrm{fst}[\alpha]).\alpha)^v)_v \\
\equiv & (z \bullet \mathrm{fst}[x.(\alpha \bullet x)]).\alpha
\end{array}
$$

$$
\begin{array}{ll}
 & ((\langle (z \bullet \mathrm{fst}[\alpha]).\alpha, (z \bullet \mathrm{snd}[\beta]).\beta\rangle)^v)_v \\
\equiv & (z \bullet \mathrm{fst}[x.(z \bullet \mathrm{snd}[y.(\langle x, y\rangle \bullet \gamma)])]).\gamma
\end{array}
$$

$$
\begin{array}{ll}
 & ((x \bullet \alpha)^v)_v \\
\equiv & x \bullet \alpha
\end{array}
$$

$$
\begin{array}{ll}
 & (([\alpha]\mathrm{not} \bullet \mathrm{not}\langle x\rangle)^v)_v \\
\equiv & [z.(z \bullet \alpha)]\mathrm{not} \bullet \mathrm{not}\langle (x \bullet \gamma).\gamma\rangle
\end{array}
$$

Figure 15: Examples of kernel terms

Figure 12. Like the CPS target calculus, the kernel is closed with respect to substitutions and reductions.

The CPS translation has a right inverse, as defined in Figure 13, which maps each term in the CPS target to the corresponding term in the kernel. Translating a term of the dual calculus into CPS and then applying the inverse CPS translation yields a corresponding term in the kernel. Figure 15 presents the five kernel terms corresponding to the five translations in Figure 14.

A term in the kernel has no (βL) or (βR) redexes. Note, however, that such redexes may be created after reduction ($\beta\neg$) is applied. In the CPS target, there is no reduction corresponding to ($\beta\neg$), but the reductions corresponding to (βL) and (βR) play a similar role.

The CPS translation on values relates in the usual way to the CPS translation on terms.

Proposition 6.4. *Let V be a value of the dual calculus. Then*

$$(V)^v \quad \equiv \quad \lambda\gamma.\,\gamma\,(V)^V.$$

The CPS translation preserves substitution of a value for a variable, and of a coterm for a covariable.

Proposition 6.5. *Let S, V, x, K, α be in the dual calculus. Then*

$$
\begin{array}{lll}
(S\{V/x\})^v & \equiv & (S)^v\{(V)^V/x\} \\
(S\{K/\alpha\})^v & \equiv & (S)^v\{(K)^v/\alpha\}.
\end{array}
$$

Applying the CPS translation followed by its inverse amounts to putting a term, coterm, or statement of the dual calculus into a normal form with regard to the reductions (βL), (βR), (ηL), (ηR), and (ς) of the source calculus.

Proposition 6.6. *Let M, K, S be terms of the dual calculus. Then*

$$
\begin{array}{lll}
M & \longrightarrow_v & ((M)^v)_v \\
K & \longrightarrow_v & ((K)^v)_v \\
S & \longrightarrow_v & ((S)^v)_v.
\end{array}
$$

In the above reductions, only the rules (βL), (βR), (ηL), (ηR), and (ς) are applied, and they are applied until they can be applied no further. (As usual, one must be careful not to apply (ηL) and (ηR) within a cut, and not to apply (ς) when the term is a variable.)

The inverse CPS translation followed by the CPS translation is the identity.

Proposition 6.7. *Let N, L, and T be in the CPS target calculus. Then*

$$
\begin{array}{lll}
((N)_v)^v & \equiv & N \\
((L)_v)^v & \equiv & L \\
((T)_v)^v & \equiv & T.
\end{array}
$$

The CPS translation preserves reductions.

Proposition 6.8. *Let M, N, K, L, S, T be in the dual calculus. Then*

$$
\left.\begin{array}{l} M \longrightarrow_v N \\ K \longrightarrow_v L \\ S \longrightarrow_v T \end{array}\right\} \ \textit{implies} \ \left\{\begin{array}{l} (M)^v \longrightarrow (N)^v \\ (K)^v \longrightarrow (L)^v \\ (S)^v \longrightarrow (T)^v. \end{array}\right.
$$

Each reduction in the dual calculus translates to zero or more reduction steps in the CPS target. In particular, reductions (ηL), (ηR), and (ς) translate to zero steps, as their left and right sides have identical CPS translations after administrative reductions have been applied.

The inverse CPS translation preserves reductions.

Proposition 6.9. *Let M, N, K, L, S, T be in the CPS target calculus. Then*

$$
\left.\begin{array}{l} M \longrightarrow N \\ K \longrightarrow L \\ S \longrightarrow T \end{array}\right\} \ \textit{implies} \ \left\{\begin{array}{l} (M)_v \longrightarrow_v (N)_v \\ (K)_v \longrightarrow_v (L)_v \\ (S)_v \longrightarrow_v (T)_v. \end{array}\right.
$$

We can summarize our results as follows. The CPS translation is a *Galois connection*; furthermore, the CPS translation followed by its inverse is the identity, so we have the stronger form of Galois connection called a *reflection*. The following is equivalent to Propositions 6.6–6.9.

Proposition 6.10. *Let M, K, S be in the dual calculus, and N, L, T be in the CPS target calculus. Then*

$$
\left.\begin{array}{l} M \longrightarrow_v (N)_v \\ K \longrightarrow_v (L)_v \\ S \longrightarrow_v (T)_v \end{array}\right\} \ \textit{iff} \ \left\{\begin{array}{ll} (M)^v \longrightarrow N & \textit{and} \quad ((N)_v)^v \equiv N \\ (K)^v \longrightarrow L & \qquad\quad ((L)_v)^v \equiv L \\ (S)^v \longrightarrow T, & \qquad\quad ((T)_v)^v \equiv T. \end{array}\right.
$$

7. CONCLUSIONS

Here is a speculation about one possible application of these ideas. Call-by-name can be inefficient because a single term may be evaluated many times. Call-by-need avoids this inefficiency by overwriting a term with its value the first time it is evaluated.

Similarly, in the dual calculus it becomes clear that call-by-value can be inefficient because a single coterm may be evaluated many times. A strategy dual to call-by-need could avoid this inefficiency by overwriting a coterm with its covalue the first time it is evaluated.

Terms in the dual calculus are not always easy to read. Compare, for instance, the λ-calculus term

$$\langle \text{fst } V, \text{snd } M \rangle$$

with a corresponding dual calculus term,

$$(V \bullet \text{fst}[x.(V \bullet \text{snd}[y.(\gamma \bullet \langle x, y \rangle)])]).\gamma$$

The latter is reminiscent of continuation-passing style — like the Pompidou Center in Paris, the plumbing is exposed on the outside. While this can make the expression harder on the eyes, it also — like CPS, and like the Pompidou Center — has the advantage of revealing structure that previously was hidden.

Call-by-name was introduced in the seminal work of Church (1932), and call-by-value was introduced in a review a few years later by Bernays (1936). Almost a half century passed between the initial publications of Church (1932) and Gentzen (1935) and their linkage in a publication by Howard (1980). After a further quarter of a century, an underlying duality between the two fundamental forms of evaluation has been revealed. What more will we discover before the centenary of the birth of λ-calculus, natural deduction, and sequent calculus?

Acknowledgements

Thanks to Pierre-Louis Curien, Olivier Danvy, Tim Griffin, Hugo Herbelin, Robert McGrail, Rex Page, Amr Sabry, Peter Selinger, Ken Shan, and Steve Zdancewic for discussions on this work.

8. 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.

George Boole (1847) The mathematical analysis of logic. Macmillan, Barclay, and Macmillan, Cambridge.

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.

H. B. Curry and R. Feys (1958) *Combinatory Logic*. North-Holland (see Chapter 9, Section E).

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.

N. G. de Bruijn (1968) The mathematical language Automath, its usage, and some of its extensions. In *Symposium on Automatic Demonstration*, Versailles, 1968, pages 29–61. Springer-Verlag, Lecture Notes in Mathematics 125, 1970.

M. Felleisen, D. Friedman, E. Kohlbecker, and B. Duba (1986) Reasoning with continuations. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 131–141, IEEE.

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

Gottlob Frege (1879) *Begriffsschrift*,a formula language, modeled upon that of arithmetic, for pure thought. Halle. Reprinted in Jan van Heijenoort, editor, *From Frege to Gödel, A Sourcebook in Mathematical Logic, 1879–1931*, Harvard University Press, 1967.

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.

Joseph Diaz Gergonne (1826) *Annales de mathématique pures et appliquées*, 16:209.

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.

Ivor Grattan Guinness (2000) *The Search for Mathematical Roots 1870–1940*. Princeton University Press.

Hugo Herbelin (1994) A lambda-calculus structure isomorphic to sequent calculus structure. In *Computer Science Logic*, pages 61–75, Springer-Verlag, LNCS 933.

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.

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

P. H. Nidditch (1969) *The Development of Mathematical Logic*. Thoemmes Press, Bristol (reprinted 1998).

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.

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.

Jean-Victor Poncelet (1818) *Annales de mathématique pures et appliquées*, 8:201.

Dag Prawitz (1965) *Natural Deduction: A Proof-Theoretical Study*. Almqvist and Wiksell, Stockholm.

John Reynolds (1993) The discoveries of continuations. *Lisp and Symbolic Computation*, 6(3/4):233–248.

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.

F. W. K. E. Schröder (1890) *Vorlesungen über die Algebra der Logik* (Teachings on the Algebra of Logic), Volume 1. Teubner Press, Leibzig.

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.

Gerald Jay Sussman and Guy Lewis Steele Jr. (1975) Scheme: an interpreter for extended lambda calculus. MIT AI Memo 319, December 1975.