Normalization in the Dual Calculus with Sigma Reductions

Jeffrey A. Vaughan Stephanie Weirich Steve Zdancewic
University of Pennsylvania
vaughan2@seas.upenn.edu {sweirich, stevez}@cis.upenn.edu

Introduction

Dual calculus (Wadler 2003, 2005) is a programming typed language whose expression syntax consists of three sorts: *terms*, *coterms*, and *statements*. Terms—which include constructors and variables—intuitively correspond to data; coterms—which include elimination forms and *covariables*—correspond to evaluation contexts. Terms have normal-forms called *values* and coterms have normal forms called *covalues*. Statements represent computations. They are composed of a term m and a coterm k, and written $m \bullet k$.

Special expressions allow variable and covariable abstraction over statements. The latter, $(m \bullet k).\alpha$, is a term corresponding to the familiar let/cc operator. The former, $x.(m \bullet k)$, is precisely dual and is called (inspired by Lovas and Crary (2006)) let/ct—"let-with-current-term." Let/cc abstractions are not values, nor are let/ct abstractions covalues.

Statements can reduce in two ways. β reductions may occur in a statement when the term's top-level constructor fits the coterm's elimination form. For instance, the rule

$$\langle m, n\rangle \bullet \mathsf{fst}[k] \to_{\beta\wedge} m \bullet k$$

states that a projection (coterm) containing sub-coterm k in juxtaposition with a pair (term) made up of m and n steps to a new statement: $m \bullet k$. Statements may also step using ς rules. These lift non-normal components of terms (dually coterms) closer to a statement's top level. For example, using ς (and reducing a resulting administrative β-redex) gives the following:

$$\langle m, n\rangle \bullet k \to^* m \bullet x.(\langle x, n\rangle \bullet k)$$

Intuitively this reduction sequence is starting to evaluate the $\langle m, n\rangle$ pair by building a stack frame ready to accept the result of computing m. Earlier work has also considered η-expansion, but we do not do so here.

Some statements may reduce by both ς and β rules. Wadler (2003) demonstrates that particular dual calculus evaluation strategies correspond to call-by-value or call-by-name evaluation in lambda calculus. Consider statement $m \bullet k$; these strategies can be summarized as follows.

call-by-value: If m is a value or let/cc, reduce by β. Otherwise, simplify m using ς-reductions.

call-by-name: If k is a covalue or let/ct, reduce by β. Otherwise, simplify k using ς-reductions.

In call-by-value dual calculus, normal-form statements always have form $v \bullet k$, where v is a value. Dually, call-by-name normal-form statements have from $m \bullet p$ where p is a co-value. These properties are desirable when dual calculus is used to model lambda calculus reduction (Wadler 2003) or lambda-mu calculus equivalences (Wadler 2005).

Wadler conjectured that dual calculus with only β rules is strongly normalizing. Dougherty et al. (2005) proved strong normalization for β-reduction with additional structural rules. Lovas and Crary (2006) formalized a proof proof of weak normalization for a similar system in the TWELF logical framework. Tzevelekos (2006) gives a reducibility candidates argument for strong normalization of dual calculus with β, η and ς reductions.

This work presents a logical relations proof of strong normalization for for call-by-value (and dually call-by-name) dual calculus with β and ς reductions. We also examine a *mixed-reduction* strategy in which each statement carries a flag indicating whether to evaluate it in a call-by-name or call-by-value fashion. We show the existence of non-terminating statements under mixed-reduction.

Dual calculus expressions may be interpreted as proofs in LK, a classical sequent calculus. (This is reminiscent of the Curry-Howard correspondence relating lambda calculus terms and natural deduction proofs.) Type assignment for terms corresponds to sequent calculus right rules, coterms correspond to left rules, and statements correspond to the cut rule.

It is trivial to show choice of reduction strategy does not effect which sequents are provable. This leads to the surprising observation that infinite-reduction sequences are not "logically harmful"—that is, do not cause logical inconsistencies—in dual calculus. This result joins others (e.g. Urban (2000)) showing that the relation between terms, types, and normalization for classical sequent systems is more subtle than for natural deduction systems studied with lambda calculus.

Dual Calculus is Strongly Normalizing

This section will sketch a logical-relations proof of strong normalization for dual calculus. Defining the logical relation and proving the main lemma will require several auxiliary definitions: language syntax, static and dynamic semantics, the logical relation itself, and a auxiliary notion of *logical substitution*.

The following grammar defines the syntax of dual calculus.

Types	A, B	$::=$	$X \mid A \wedge A \mid A \vee A \mid \neg A$
Terms	m, n	$::=$	$x \mid \langle m, n\rangle \mid [k]\mathsf{not} \mid \langle m\rangle\mathsf{inl} \mid \langle n\rangle\mathsf{inr} \mid (S).\alpha$
Coterms	k, l	$::=$	$\alpha \mid \mathsf{fst}[k] \mid \mathsf{snd}[l] \mid \mathsf{not}\langle m\rangle \mid [k, l] \mid x.(S)$
Statements	S	$::=$	$m \bullet k$

Some terms are also values. Intuitively values are terms in which all let/ct subterms are "suspended" by an enclosing not. Formally the set of call-by-value values is generated by the following grammar.

$$\text{Values} \quad v, w \quad ::= x \mid \langle v, w\rangle \mid [k]\mathsf{not} \mid \langle v\rangle\mathsf{inl} \mid \langle w\rangle\mathsf{inr}$$

The dynamic semantics for call-by-value dual calculus uses term evaluation contexts, written E. Term evaluation contexts are defined by

$$E ::= \langle\{\}, n\rangle \mid \langle v, \{\}\rangle \mid \langle\{\}\rangle\mathsf{inl} \mid \langle\{\}\rangle\mathsf{inr}$$

where n is not be a value. The symbol $\{\}$ represents a hole which may be filled by a term. $E\{m\}$ denotes the term created by replacing E's hole with m.

The call-by-value dynamic semantics are as follows.

$$\langle v, w\rangle \bullet \mathsf{fst}[k] \to_\beta v \bullet k \qquad \langle v, w\rangle \bullet \mathsf{snd}[l] \to_\beta w \bullet l$$

$$\langle v\rangle\mathsf{inl} \bullet [k, l] \to_\beta v \bullet k \qquad \langle w\rangle\mathsf{inr} \bullet [k, l] \to_\beta w \bullet l$$

$$[k]\mathsf{not} \bullet \mathsf{not}\langle m\rangle \to_\beta m \bullet k$$

$$v \bullet x.(S) \to_\beta \{v/x\}S \qquad (S).\alpha \bullet k \to_\beta \{k/\alpha\}S$$

$$E\{m\} \bullet k \to_\varsigma (m \bullet x.(E\{x\} \bullet \beta)).\beta \bullet k$$

We use a logical-relations argument to show that call-by-value dual calculus is strongly normalizing. The following four indexed sets represent unary relations on dual calculus expressions. The main lemma will demonstrate that all well-typed expressions are logical—that is, are members of a relation. Strong normalization follows as a corollary.

$$\begin{array}{rcl}
S \in S[\![\#]\!] & \text{iff} & S \to^* S' \not\to \\
\\
x \in V[\![A]\!] & & \text{(always)} \\
\langle v, w\rangle \in V[\![A \wedge B]\!] & \text{iff} & v \in V[\![A]\!] \text{ and } w \in V[\![B]\!] \\
\langle v\rangle\mathsf{inl} \in V[\![A \vee B]\!] & \text{iff} & v \in V[\![A]\!] \\
\langle w\rangle\mathsf{inr} \in V[\![A \vee B]\!] & \text{iff} & w \in V[\![B]\!] \\
[k]\mathsf{not} \in V[\![\neg A]\!] & \text{iff} & k \in K[\![A]\!] \\
\\
k \in K[\![A]\!] & \text{iff} & \text{for all } v \in V[\![A]\!], v \bullet k \in S[\![\#]\!] \\
\\
m \in M[\![A]\!] & \text{iff} & \text{for all } k \in K[\![A]\!], m \bullet k \in S[\![\#]\!]
\end{array}$$

We can make several observations about this family of relations. First, they are defined on open terms, without closing substitutions. This approach is useful because dual calculus, like LK, has no closed well-typed terms. Second, $V[\![A]\!]$ and $K[\![A]\!]$ are defined by mutual recursion on their type parameters, however the "recursive call" in $K[\![A]\!]$ does not mention a structurally smaller type. Instead, the definition of $V[\![A]\!]$ must be unfolded to verify that the recursion is well-founded. Third, it is while type indexed, the relation family is otherwise oblivious to types; it does not mention a context, and contains expressions which cannot be typed in any context. This is useful as the main lemma does not depend on subject reduction (a property dual calculus enjoys, but for which we know no published proof). Fourth, $M[\![\cdot]\!]$ is defined by universally quantifying over $K[\![\cdot]\!]$, and $K[\![\cdot]\!]$ by universally quantifying over $V[\![\cdot]\!]$. This strikingly similar to Pitts's (2005) TT-closure method for reasoning about program equivalence.

Before stating the main lemma, we need two more definitions.

Space constraints prevent us from giving the static semantics for dual calculus. We follow the presentation of Wadler (2005), which simplifies the original system (Wadler 2003) by liberalizing the cut and identity rules and by removing the now superfluous structural rules. Dual calculus is typed using three mutually inductive relations. Each includes an *antecedent* context Γ and a *succedent* context Θ. Antecedent contexts map variables to types, and succedent contexts map covariables to types. For call-by-value we can read the typing judgments as follows.

- Right Sequent, $\Gamma \vdash m\colon A; \Theta$. Term m has type A in contexts Γ and Θ. Intuitively, evaluating m might yield a return value at type A or it might throw to a continuation described by Θ.
- Left Sequent, $\Gamma; k\colon A \vdash \Theta$. Coterm k is a continuation which can accept a value of type A.
- Center Sequent, $S : (\Gamma \vdash \Theta)$. Statement S is a computation which takes one value per binding in Γ as input. S yields only a single output, whose type is included in Θ.

Substitutions are partial maps that take variables to terms and covariables to coterms. Some substitutions have the special property of replacing (co)variables with logical (co)terms. We say substitution σ is Γ, Θ-logical when

(i) for all $x \in \mathrm{dom}(\sigma)$, $\sigma(x) \in V[\![\Gamma(x)]\!]$, and

(ii) for all $\alpha \in \mathrm{dom}(\sigma)$, $\sigma(\alpha) \in K[\![\Theta(\alpha)]\!]$.

Lemma (Main Lemma)**.** *Suppose $\mathcal{D}$ is a typing derivation, then*

- *if $\mathcal{D} :: S\colon (\Gamma \vdash \Theta)$ then for all Γ, Θ-logical σ, $\sigma(S) \in S[\![\#]\!]$,*
- *if $\mathcal{D} :: \Gamma \vdash v\colon A; \Theta$ then for all Γ, Θ-logical σ, $\sigma(v) \in V[\![A]\!]$,*
- *if $\mathcal{D} :: \Gamma \vdash m\colon A; \Theta$ then for all Γ, Θ-logical σ, $\sigma(m) \in M[\![A]\!]$,*
- *if $\mathcal{D} :: \Gamma; k\colon A \vdash \Theta$ then for all Γ, Θ-logical σ, $\sigma(k) \in K[\![A]\!]$.*

Proof Sketch. By induction on the structure on the $\mathcal{D}$. Note the lemma deeply nests the quantification on σ. This is required to get a strong induction hypothesis for the $\mathcal{D} :: \Gamma \vdash (S).\alpha\colon A; \Theta$ case. Many cases use the following fact: if $S \to S'$ and $S' \in S[\![\#]\!]$ then $S \in S[\![\#]\!]$. The full proof also requires one-off lemmas showing that various sigma reductions involving elements of $M[\![\cdot]\!]$, $V[\![\cdot]\!]$ and $K[\![\cdot]\!]$ produce elements of $K[\![\cdot]\!]$. $\square$

Corollary (Strong normalization)**.** *Every well-typed dual calculus statement is strongly normalizing.*

Proof. Suppose $S\colon (\Gamma \vdash \Theta)$, then $S \in S[\![\#]\!]$ by the main lemma. Hence $S \to^* S' \not\to$. $\square$

Mixed Reduction is Not Strongly Normalizing

Mixed reduction is a natural variant of dual calculus. In mixed reduction every statement carries a tag indicating whether its top-level redex should reduce using call-by-name or call-by-value rules. For instance $m \overset{\leftarrow}{\bullet} k \to_{mixed} S$ when $m \bullet k \to_\beta S$ in call-by-value. (And similarly $m \overset{\rightarrow}{\bullet} k$ reduces when there a suitable call-by-name reduction.) The ς-reductions include

$$E\{m\} \overset{\leftarrow}{\bullet} k \to (m \overset{\leftarrow}{\bullet} x.(E\{x\} \overset{\leftarrow}{\bullet} \beta)).\beta \overset{\leftarrow}{\bullet} k$$

and its dual.

While call-by-value (dually call-by-name) dual calculus is both strongly normalizing and deterministic, mixed reduction is not strongly normalizing. For a counterexample, the statement $z \overset{\rightarrow}{\bullet} [\alpha, x.(\langle y, x\rangle \overset{\leftarrow}{\bullet} \beta)]$ reduces to itself in six steps.

References

Daniel J. Dougherty, Silvia Ghilezan, Pierre Lescanne, and Silvia Likavec. Strong normalization of the dual classical sequent calculus. In *LPAR*, pages 169–183, 2005.

William Lovas and Karl Crary. Structural normalization for classical natural deduction. Available from `http://www.cs.cmu.edu/~wlovas/papers/clnorm.pdf`, 2006.

A. M. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. The MIT Press, 2005. ISBN 0-262-16228-8.

Nikos Tzevelekos. Investigations on the dual calculus. *Theor. Comput. Sci.*, 360(1):289–326, 2006. ISSN 0304-3975.

Christian Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

Philip Wadler. Call-by-value is dual to call-by-name. In *ICFP '03*, Uppsala, Sweden, 2003.

Philip Wadler. Call-by-value is dual to call-by-name, reloaded. In *Rewriting Techniques and Applications '05*, Nara, 2005.