Strong normalization of the dual classical sequent calculus

Daniel Dougherty,[1] Silvia Ghilezan,[2] Pierre Lescanne[3] and Silvia Likavec[2,4]

[1] Worcester Polytechnic Institute, USA, dd@wpi.edu
[2] Faculty of Engineering, University of Novi Sad, Serbia, gsilvia@uns.ns.ac.yu
[3] ENS Lyon, France, pierre.lescanne@ens-lyon.fr
[4] Dipartimento di Informatica, Università di Torino, Italy, likavec@di.unito.it

Abstract. We investigate some syntactic properties of Wadler's dual calculus, a term calculus which corresponds to classical sequent logic in the same way that Parigot's $\lambda\mu$ calculus corresponds to classical natural deduction. Our main result is strong normalization theorem for reduction in the dual calculus; we also prove some confluence results for the typed and untyped versions of the system.

1 Introduction

This paper establishes some of the key properties of reduction underlying Wadler's dual calculus [30, 31]. The basic system, obtained as a term-assignment system for classical sequent calculus, is not confluent, inheriting the well-known anomaly of classical cut-elimination. Wadler recovers confluence by restricting to reduction strategies corresponding to (either of) the call-by-value or call-by-name disciplines, indeed these subcalculi and the duality between them are the main focus of attention in Wadler's work.

In this paper we are less interested in call-by-value and call-by-name *per se* than in the pure combinatorics of reduction itself, consequently we work with as few restrictions as possible on the system. We prove strong normalization (SN) for unrestricted reduction of typed terms, including expansion rules capturing extensionality. We show that once the obvious obstacle to confluence is removed (the "critical pair" in the reduction system) confluence holds in both the typed and untyped versions of the term calculus. This critical pair (see Section 3) can be disambiguated in two ways but the proof we give dualizes to yield confluence results for each system, an example of the "two theorems for the price of one" benefit of duality.

The dual calculus is an embodiment of the "proofs-as-programs" paradigm in the setting of classical logic, as well as being a clear expression of the relationship between call-by-name and call-by-value in functional programming. So the fundamental syntactic results given here should play an important role in the currently active investigations into the relationship between classical logic and computation.

Background The Curry-Howard correspondence expresses a fundamental connection between logic and computation [18]. In its traditional form, terms in the λ-calculus encode proofs in intuitionistic natural deduction; from another perspective the proofs serve as typing derivations for the terms. Griffin extended the Curry-Howard correspondence

to classical logic in his seminal 1990 POPL paper [16], by observing that classical tautologies suggest typings for certain control operators. This initiated a vigorous line of research: on the one hand classical calculi can be seen as pure programming languages with explicit representations of control, while at the same time terms can be tools for extracting the constructive content of classical proofs [21, 3]. In particular the $\lambda\mu$ calculus of Parigot [23] has been the foundation of a number of investigations [24, 11, 22, 5, 1] into the relationship between classical logic and theories of control in programming languages.

As early as 1989 Filinsky [14] explored the notion that the reduction strategies call-by-value and call-by-name were dual to each other. Filinski defined a symmetric lambda-calculus in which values and continuations comprised distinct syntactic sorts and whose denotational semantics expressed the call-by-name vs call-by-value duality in a precise categorical sense. Later Selinger [27] modeled the call-by-name and call-by-value variants of the $\lambda\mu$ by dual control and co-control categories.

These two lines of investigation come together nicely in the framework of classical *sequent calculus*. In contrast to natural deduction proof systems (upon which Parigot's $\lambda\mu$, for example, is based) sequent calculi exhibit inherent symmetries not just at the level of terms, but of proof structures as well. There are several term calculi based on sequent calculus, the most relevant to the current study are those in which terms unambiguously encode sequent derivations ([29, 9, 19, 2] for example) in which reduction corresponds to cut elimination. Herbelin [17], and subsequently Curien and Herbelin [9] defined the system $\overline{\lambda}\mu\widetilde{\mu}$, a sequent calculus-inspired calculus exhibiting symmetries in the syntax, whose terms represent derivations in the implicational fragment of Gentzen's system LK [15]. In addition, as described in [9], the sequent calculus basis for $\overline{\lambda}\mu\widetilde{\mu}$ supports an interpretation of the reduction rules of the system as operations of an abstract machine. In particular, the right- and left-hand sides of a sequent directly represent the *code* and *environment* components of the machine. This perspective is elaborated more fully in [8]. See [7] for a discussion of the importance of symmetries in computation. In [2], a calculus, which interprets directly the implicational sequent logic, is proposed as a language in which many kinds of other calculi can be implemented, from λ-calculus to $\overline{\lambda}\mu\widetilde{\mu}$ through a calculus of explicit substitution and $\lambda\mu$.

The Symmetric Lambda Calculus of Barbanera and Berardi [3], although not based on sequent calculus, belongs in the tradition of exploiting the symmetries found in classical logic, in their case with the goal of extracting constructive content from classical proofs. Barbanera and Berardi [3] proved SN for their calculus using a "symmetric candidates" technique; Urban and Bierman [29] adapted their technique to prove SN for their sequent-based system; Lengrand [19] shows how simply-typed $\overline{\lambda}\mu\widetilde{\mu}$ and the calculus of Urban and Bierman [29] are mutually interpretable, so that the strong normalization proof of the latter calculus yields another proof of strong normalization for simply-typed $\overline{\lambda}\mu\widetilde{\mu}$. Polonovski [25] presents a proof of SN for $\overline{\lambda}\mu\widetilde{\mu}$ with explicit substitutions using the symmetric candidates idea. Pym and Ritter [26] identify two forms of disjunction for Parigot's[23] $\lambda\mu$ calculus; they prove strong normalization for $\lambda\mu\nu$ calculus ($\lambda\mu$ calculus extended with such disjunction). David and Nour [10] give an arithmetical proof of strong normalization for a symmetric $\lambda\mu$ calculus.

The dual calculus Wadler's dual calculus [30] refines and unifies these themes. It is a term-assignment system based on classical sequent calculus, and a key step is that implication is not taken as a primitive connective. It turns out that this permits a very clear expression of the way in which the traditional duality between the left- and right-hand sides of a sequent reflects the duality between call-by-value and call-by-name.

Unfortunately these beautiful symmetries come at the price of some anomalies in the behavior of reduction. The unrestricted reduction relation in the dual calculus (as well as in $\overline{\lambda}\mu\widetilde{\mu}$) has a critical pair, and indeed this system is not confluent. In [30] Wadler gives two restricted versions of each reduction rule obtaining subcalculi which naturally correspond to call-by-value and call-by-name, respectively. He then defines translations of these systems into the simply-typed λ-calculus; each translation both preserves and reflects reductions. See Propositions 6.6, 6.9, 6.10 on [30]. (Curien and Herbelin [9] gave a similar encoding of their $\overline{\lambda}\mu\widetilde{\mu}$ calculus.)

It was "claimed without proof" in [30], that these call-by-value and call-by-name reductions are confluent and that the call-by-value and call-by-name reduction relations (without expansions) are strongly normalizing. But in fact confluence and strong normalization for each of call-by-value and call-by-name follows from the corresponding results in the λ-calculus by diagram-chasing through the CPS translations into the simply-typed λ-calculus, given the fact that reductions are preserved and reflected by the translations.

In [31] the emphasis is on the equational theory of the dual calculus. The equations of the dual calculus include a group of equations called "η-equations" which express extensionality properties; these equations play an important role in the relationship between the dual calculus and $\lambda\mu$. The relationship with Parigot's $\lambda\mu$ is worked out, the result is a clear notion of duality for $\lambda\mu$.

Summary of results

We prove that unrestricted reduction of typed expressions in the dual calculus is strongly normalizing. The proof is a variation on the "semantical" method of reducibility, where types are interpreted as *pairs* of sets of terms (observe: yet another symmetry). Our proof technique uses a fixed-point construction similar to that in [3] but the technique is considerably simplified here (Section 6).

In fact our proof technique also shows the strong normalization for the reduction system including the η-expansion rules of the dual calculus. Due to space restrictions we only outline the treatment of the expansions but the machinery is the same as for the core calculus and filling in the missing details should only be an exercise for the reader.

To our knowledge none of the previous treatments of strong normalization for classical calculi has addressed extensionality rules.

We prove that if we disambiguate the single critical pair in the system, by giving priority to either the "left" or to the "right" reductions, the resulting subsystems are confluent. Furthermore reduction is confluent whether terms are typed or untyped. The proof is an application of Takahashi's parallel reductions technique [28]; we prove the result for one system and are able to conclude the result for the other by duality (Section 4).

The relationship between our results and those in [30, 31] is somewhat subtle. Wadler is motivated by programming language concerns and so is led to focus on sub-calculi of the dual calculus corresponding to call-by-name and call-by-value reduction; not only is the critical pair in the system removed but reductions must act on "values" (or "covalues"). In contrast, we are interested in the pure combinatorics of reduction, and so

- in exploring strong normalization we consider unrestricted reduction of typed terms (as well as incorporating expansions), and

- in exploring confluence we consider reduction of untyped terms, and impose only the restriction that the critical pair (which demonstrably destroys confluence) be disambiguated.

2 Syntax

Following Wadler, we distinguish three syntactic categories: *terms*, *coterms*, and *statements*. Terms yield values, while coterms consume values. A statement is a cut of a term against a coterm. We call the expressions in the union of these three categories *D-expressions*.

Let r, q range over the set Λ_{R} of terms, e, f range over the set Λ_{L} of coterms, and c ranges over statements. Then the syntax of the dual calculus is given by the following:

$$\begin{array}{lll} \textsf{Term:} & r,q & ::= x \mid \langle r,\, q\rangle \mid \langle r\rangle\mathrm{inl} \mid \langle r\rangle\mathrm{inr} \mid [e]\mathrm{not} \mid \mu\alpha\,.\,c \\ \textsf{Coterm:} & e,f & ::= \alpha \mid [e,\, f] \mid \mathrm{fst}[e] \mid \mathrm{snd}[e] \mid \mathrm{not}\langle r\rangle \mid \widetilde{\mu}x\,.\,c \\ \textsf{Statement:} & c & ::= (\!|\, r \bullet e \,|\!) \end{array}$$

where x ranges over a set of term variables Var_{R}, $\langle r,\, q\rangle$ is a pair, $\langle r\rangle\mathrm{inl}$ ($\langle r\rangle\mathrm{inr}$) is an injection on the left (right) of the sum, $[e]\mathrm{not}$ is a complement of a coterm, and $\mu\alpha\,.\,c$ is a covariable abstraction. Next, α ranges over a set of covariables Var_{L}, $[e,\, f]$ is a case, $\mathrm{fst}[e]$ ($\mathrm{snd}[e]$) is a projection from the left (right) of a product, $\mathrm{not}\langle r\rangle$ is a complement of a term, and $\widetilde{\mu}x\,.\,c$ is a variable abstraction. Finally $(\!|\, r \bullet e \,|\!)$ is a cut. The term variables can be bound by μ abstraction, whereas the coterm variables can be bound by $\widetilde{\mu}$ abstraction. The sets of free term and coterm variables, Fv_{R} and Fv_{L}, are defined as usual, respecting Barendregt's convention [4] that no variable can be both, bound and free, in the expression. As in [30, 31], angle brackets always surround terms and square brackets always surround coterms. Also, curly brackets are used for substitution and to denote holes in contexts.

We decided to slightly alter the notation given by Wadler. First of all, we use $\mu\alpha\,.\,c$ and $\widetilde{\mu}x\,.\,c$ instead of $(S).\alpha$ and $x.(S)$. Furthermore, we use $(\!|\, r \bullet e \,|\!)$ for statements, since from our point of view it is easier to read than $r \bullet e$. Finally, the lowercase letters that we use to denote D-expressions should help to distinguish such expressions from types.

3 Reduction rules

Wadler defines the dual calculus, giving the reductions that respect call-by-value and call-by-name reduction strategies, respectively. We give the reduction rules for an unrestricted calculus in Figure 1. Of course the notion of reduction is defined on raw expressions, and does not make use of any typing constrains. We use $\twoheadrightarrow$ to denote the

$$
\begin{array}{lll}
(\beta\widetilde{\mu}) & \langle\!| \, r \bullet \widetilde{\mu}x \, . \, c \, |\!\rangle & \to c\{r/x\} \\
(\beta\mu) & \langle\!| \, \mu\alpha \, . \, c \bullet e \, |\!\rangle & \to c\{e/\alpha\} \\
(\beta\wedge) & \langle\!| \, \langle r, \, q\rangle \bullet \mathrm{fst}[e] \, |\!\rangle & \to \langle\!| \, r \bullet e \, |\!\rangle \\
(\beta\wedge) & \langle\!| \, \langle r, \, q\rangle \bullet \mathrm{snd}[e] \, |\!\rangle & \to \langle\!| \, q \bullet e \, |\!\rangle \\
(\beta\vee) & \langle\!| \, \langle r\rangle\mathrm{inl} \bullet [e, \, f] \, |\!\rangle & \to \langle\!| \, r \bullet e \, |\!\rangle \\
(\beta\vee) & \langle\!| \, \langle r\rangle\mathrm{inr} \bullet [e, \, f] \, |\!\rangle & \to \langle\!| \, r \bullet f \, |\!\rangle \\
(\beta\neg) & \langle\!| \, [e]\mathrm{not} \bullet \mathrm{not}\langle r\rangle \, |\!\rangle & \to \langle\!| \, r \bullet e \, |\!\rangle
\end{array}
$$

Fig. 1. Reduction rules for the dual calculus

reflexive transitive closure of $\to$ (with a similar convention for other relations denoted by other arrows).

Remark 1. The following observation will be useful later; it is the analogue of the standard λ-calculus trick of "promoting head reductions." Specifically, if a reduction sequence out of a statement ever does a top-level μ-reduction, then we can promote the first such reduction to be the first in the sequence, in the following sense: the reduction sequence $\langle\!| \, \mu\alpha.c \bullet e \, |\!\rangle \longrightarrow\!\!\!\!\to \langle\!| \, \mu\alpha.c' \bullet e' \, |\!\rangle \longrightarrow c'\{e'/\alpha\}$ can be transformed to the reduction sequence $\langle\!| \, \mu\alpha.c \bullet e \, |\!\rangle \longrightarrow c\{e/\alpha\} \longrightarrow\!\!\!\!\to c'\{e'/\alpha\}$.

The calculus has a critical pair $\langle\!| \, \mu\alpha \, . \, c_1 \bullet \widetilde{\mu}x \, . \, c_2 \, |\!\rangle$ where both the $(\beta\widetilde{\mu})$ and $(\beta\mu)$ rules can be applied ambiguously, producing two different results. For example,

$$\langle\!| \, \mu\alpha.\langle\!| \, y \bullet \beta \, |\!\rangle \bullet \widetilde{\mu}x.\langle\!| \, z \bullet \gamma \, |\!\rangle \, |\!\rangle \to \langle\!| \, y \bullet \beta \, |\!\rangle, \qquad \langle\!| \, \mu\alpha.\langle\!| \, y \bullet \beta \, |\!\rangle \bullet \widetilde{\mu}x.\langle\!| \, z \bullet \gamma \, |\!\rangle \, |\!\rangle \to \langle\!| \, z \bullet \gamma \, |\!\rangle$$

Hence, the calculus is not confluent. But if the priority is given to one of the rules, we obtain two subcalculi $\mathsf{Dual}_{\mathrm{R}}$ and $\mathsf{Dual}_{\mathrm{L}}$. Therefore, there are two possible reduction strategies in the dual calculus that depend on the orientation of the critical pair. The system $\mathsf{Dual}_{\mathrm{L}}$ with call-by-value reduction is obtained if the priority is given to (μ) redexes, whereas the system $\mathsf{Dual}_{\mathrm{R}}$ with call-by-name reduction is obtained by giving the priority to $(\widetilde{\mu})$ redexes.

That is, $\mathsf{Dual}_{\mathrm{R}}$ is defined by refining the reduction rule $(\beta\mu)$ as follows

$$\langle\!| \, \mu\alpha.c \bullet \underline{e} \, |\!\rangle \to c\{\underline{e}/\alpha\} \qquad \text{provided } \underline{e} \text{ is a coterm not of the form } \widetilde{\mu}x.c'$$

and $\mathsf{Dual}_{\mathrm{L}}$ is defined similarly by refining the reduction rule $(\beta\widetilde{\mu})$ as follows

$$\langle\!| \, \underline{r} \bullet \widetilde{\mu}x \, . \, c \, |\!\rangle \to c\{\underline{r}/x\} \qquad \text{provided } \underline{r} \text{ is a term not of the form } \mu\alpha.c'$$

Both systems $\mathsf{Dual}_{\mathrm{R}}$ and $\mathsf{Dual}_{\mathrm{L}}$ are shown to be confluent in Section 4.

Implication, λ-terms, and application

Implication can be defined in terms of other connectives, indeed in two ways:

- under call-by-value $A \supset B \equiv \neg(A \wedge \neg B)$
- under call-by-name $A \supset B \equiv \neg A \vee B$.

Under each of these conventions we can define expressions $\lambda x.r$ and $q@e$ validating the reduction $\langle\!| \, \lambda x \, . \, r \bullet q@e \, |\!\rangle \quad \to \quad \langle\!| \, q \bullet \widetilde{\mu}x.\langle\!| \, r \bullet e \, |\!\rangle \, |\!\rangle$ in the sense that when $\supset$ is defined by call-by-value and the translation of $\langle\!| \, \lambda x.r \bullet q@e \, |\!\rangle$ is reduced according to the call-by-value calculus, we get to $\langle\!| \, q \bullet \widetilde{\mu}x.\langle\!| \, r \bullet e \, |\!\rangle \, |\!\rangle$ after several steps (and the same claim holds for call-by-name).

4 Confluence of the dual calculus

To prove the confluence of the dual calculi $\mathsf{Dual}_{\mathrm{R}}$ and $\mathsf{Dual}_{\mathrm{L}}$ we adopt the technique of parallel reductions given by Takahashi in [28] (see also [20]). This approach consists of simultaneously reducing all the redexes existing in an expression and is simpler than standard Tait-and-Martin-Löf proof of confluence of β-reduction for lambda calculus. We omit the proofs for the lack of space. The detailed proofs of confluence for $\widetilde{\lambda\mu\widetilde{\mu}}$ can be found in [20].

We denote the union of all the reduction relations for $\mathsf{Dual}_{\mathrm{R}}$ by $\underset{R}{\longrightarrow}$. Its reflexive transitive closure and closure by congruence is denoted by $\underset{R}{\twoheadrightarrow}$.

First, we define the notion of parallel reduction $\Rightarrow_{\mathrm{R}}$ for $\mathsf{Dual}_{\mathrm{R}}$. Since we will show that $\underset{R}{\twoheadrightarrow}$ is the reflexive and transitive closure of $\Rightarrow_{\mathrm{R}}$, in order to prove the confluence of $\underset{R}{\twoheadrightarrow}$ it is enough to prove the diamond property for $\Rightarrow_{\mathrm{R}}$. The diamond property for $\Rightarrow_{\mathrm{R}}$ follows from the stronger "Star property" for $\Rightarrow_{\mathrm{R}}$ that we prove.

Applying the duality transformations that Wadler gives, reductions dualize as well, and in particular a μ-step is dual to a $\widetilde{\mu}$-step. A reduction from s to t under the restriction that μ-steps have priority over $\widetilde{\mu}$-steps dualizes to a reduction from the dual of s to the dual of t under the restriction that $\widetilde{\mu}$-steps have priority over μ-steps. So if we prove confluence for one of these systems, we get confluence for the other by diagram-chasing a duality argument.

4.1 Parallel reduction for $\mathsf{Dual}_{\mathrm{R}}$

The notion of parallel reduction is defined directly by induction on the structure of D-expressions, and does not need the notion of residual or any other auxiliary notion.

Definition 2 (Parallel reduction for $\mathsf{Dual}_{\mathrm{R}}$). *The parallel reduction, denoted by $\Rightarrow_R$ is defined inductively in Figure 2, where $\underline{e}$ is a coterm not of the form $\widetilde{\mu}x.c'$.*

Lemma 3. *For every D-expression D, $D \Rightarrow_R D$.*

Lemma 4 (Substitution lemma). *If $x \neq y$ and $x \notin Fv_R(r_2)$ then*

1. $D\{r_1/x\}\{r_2/y\} = D\{r_2/y\}\{r_1\{r_2/y\}/x\}$;
2. $D\{e/\alpha\}\{r/x\} = D\{r/x\}\{e\{r/x\}/\alpha\}$;
3. $D\{r/x\}\{e/\alpha\} = D\{e/\alpha\}\{r\{e/\alpha\}/x\}$;
4. $D\{e_1/\alpha\}\{e_2/\beta\} = D\{e_2/\beta\}\{e_1\{e_2/\beta\}/\alpha\}$.

Lemma 5.

1. *If $D \underset{R}{\longrightarrow} D'$ then $D \Rightarrow_R D'$;*
2. *If $D \Rightarrow_R D'$ then $D \underset{R}{\twoheadrightarrow} D'$;*
3. *If $D \Rightarrow_R D'$ and $H \Rightarrow_R H'$, then $D\{H/x\} \Rightarrow_R D'\{H'/x\}$ and $D\{H/\alpha\} \Rightarrow_R D'\{H'/\alpha\}$.*

From the points 1. and 2. in Lemma 5 we conclude that $\underset{R}{\twoheadrightarrow}$ is the reflexive and transitive closure of $\Rightarrow_{\mathrm{R}}$.

$$\frac{}{x \Rightarrow_{\mathrm{R}} x}\ (pr1_R) \qquad \frac{c \Rightarrow_{\mathrm{R}} c'}{\mu\alpha.\, c \Rightarrow_{\mathrm{R}} \mu\alpha.\, c'}\ (pr2_R) \qquad \frac{}{\alpha \Rightarrow_{\mathrm{R}} \alpha}\ (pr3_R) \qquad \frac{c \Rightarrow_{\mathrm{R}} c'}{\widetilde{\mu}x.\, c \Rightarrow_{\mathrm{R}} \widetilde{\mu}x.\, c'}\ (pr4_R)$$

$$\frac{r \Rightarrow_{\mathrm{R}} r', q \Rightarrow_{\mathrm{R}} q'}{\langle r,\, q\rangle \Rightarrow_{\mathrm{R}} \langle r',\, q'\rangle}\ (pr5_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r'}{\langle r\rangle \mathrm{inl} \Rightarrow_{\mathrm{R}} \langle r'\rangle \mathrm{inl}}\ (pr6_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r'}{\langle r\rangle \mathrm{inr} \Rightarrow_{\mathrm{R}} \langle r'\rangle \mathrm{inr}}\ (pr7_R)$$

$$\frac{e \Rightarrow_{\mathrm{R}} e', f \Rightarrow_{\mathrm{R}} f'}{[e,\, f] \Rightarrow_{\mathrm{R}} [e',\, f']}\ (pr8_R) \qquad \frac{e \Rightarrow_{\mathrm{R}} e'}{\mathrm{fst}[e] \Rightarrow_{\mathrm{R}} \mathrm{fst}[e']}\ (pr9_R) \qquad \frac{e \Rightarrow_{\mathrm{R}} e'}{\mathrm{snd}[e] \Rightarrow_{\mathrm{R}} \mathrm{snd}[e']}\ (pr10_R)$$

$$\frac{r \Rightarrow_{\mathrm{R}} r'}{\mathrm{not}\langle r\rangle \Rightarrow_{\mathrm{R}} \mathrm{not}\langle r'\rangle}\ (pr11_R) \qquad \frac{e \Rightarrow_{\mathrm{R}} e'}{[e]\mathrm{not} \Rightarrow_{\mathrm{R}} [e']\mathrm{not}}\ (pr12_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r', e \Rightarrow_{\mathrm{R}} e'}{(\!|\, r \bullet e \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, r' \bullet e' \,|\!)}\ (pr13_R)$$

$$\frac{c \Rightarrow_{\mathrm{R}} c', \underline{e} \Rightarrow_{\mathrm{R}} \underline{e'}}{(\!|\, \mu\alpha.\, c \bullet \underline{e} \,|\!) \Rightarrow_{\mathrm{R}} c'\{\underline{e'}/\alpha\}}\ (pr14_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r', c \Rightarrow_{\mathrm{R}} c'}{(\!|\, r \bullet \widetilde{\mu}x.\, c \,|\!) \Rightarrow_{\mathrm{R}} c'\{r'/x\}}\ (pr15_R)$$

$$\frac{r \Rightarrow_{\mathrm{R}} r', q \Rightarrow_{\mathrm{R}} q', e \Rightarrow_{\mathrm{R}} e'}{(\!|\, \langle r,\, q\rangle \bullet \mathrm{fst}[e] \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, r' \bullet e' \,|\!)}\ (pr16_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r', q \Rightarrow_{\mathrm{R}} q', e \Rightarrow_{\mathrm{R}} e'}{(\!|\, \langle r,\, q\rangle \bullet \mathrm{snd}[e] \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, q' \bullet e' \,|\!)}\ (pr17_R)$$

$$\frac{r \Rightarrow_{\mathrm{R}} r', e \Rightarrow_{\mathrm{R}} e', f \Rightarrow_{\mathrm{R}} f'}{(\!|\, \langle r\rangle \mathrm{inl} \bullet [e,\, f] \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, r' \bullet e' \,|\!)}\ (pr18_R) \qquad \frac{r \Rightarrow_{\mathrm{R}} r', e \Rightarrow_{\mathrm{R}} e', f \Rightarrow_{\mathrm{R}} f'}{(\!|\, \langle r\rangle \mathrm{inr} \bullet [e,\, f] \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, r' \bullet f' \,|\!)}\ (pr19_R)$$

$$\frac{r \Rightarrow_{\mathrm{R}} r', e \Rightarrow_{\mathrm{R}} e'}{(\!|\, [e]\mathrm{not} \bullet \mathrm{not}\langle r\rangle \,|\!) \Rightarrow_{\mathrm{R}} (\!|\, r' \bullet e' \,|\!)}\ (pr20_R)$$

Fig. 2. Parallel reduction

4.2 Confluence of $\mathsf{Dual}_{\mathrm{R}}$

Next, we define the D-expression D^* which is obtained from D by simultaneously reducing all the existing redexes of the D-expression D.

Definition 6. *Let D be an arbitrary D-expression of Dual_R. The D-expression D^* is defined inductively as follows:*

$(*1_R)\ x^* \equiv x \qquad (*2_R)\ (\mu\alpha.\, c)^* \equiv \mu\alpha.\, c^* \qquad (*3_R)\ \alpha^* \equiv \alpha \qquad (*4_R)\ (\widetilde{\mu}x.\, c)^* \equiv \widetilde{\mu}x.\, c^*$

$(*5_R)\ \langle r,\, q\rangle^* \equiv \langle r^*,\, q^*\rangle \quad (*6_R)\ \langle r\rangle\mathrm{inl}^* \equiv \langle r^*\rangle\mathrm{inl} \quad (*7_R)\ \langle r\rangle\mathrm{inr}^* \equiv \langle r^*\rangle\mathrm{inr}$

$(*8_R)\ [e,\, f]^* \equiv \langle e^*,\, f^*\rangle \quad (*9_R)\ \mathrm{fst}[e]^* \equiv \mathrm{fst}[e^*] \quad (*10_R)\ \mathrm{snd}[e]^* \equiv \mathrm{snd}[e^*]$

$(*11_R)\ \mathrm{not}\langle r\rangle^* \equiv \mathrm{not}\langle r^*\rangle \quad (*12_R)\ [e]\mathrm{not}^* \equiv [e^*]\mathrm{not}$

$(*13_R)\ (\!|\, r \bullet e \,|\!)^* \equiv (\!|\, r^* \bullet e^* \,|\!)$ *if* $(\!|\, r \bullet e \,|\!) \neq (\!|\, [e']\mathrm{not} \bullet \mathrm{not}\langle r'\rangle \,|\!)$ *and*
$(\!|\, r \bullet e \,|\!) \neq (\!|\, \mu\alpha.\, c \bullet \underline{e} \,|\!)$ *and* $(\!|\, r \bullet e \,|\!) \neq (\!|\, r \bullet \widetilde{\mu}x.\, c \,|\!)$ *and*
$(\!|\, r \bullet e \,|\!) \neq (\!|\, \langle r',\, q\rangle \bullet \mathrm{fst}[e'] \,|\!)$ *and* $(\!|\, r \bullet e \,|\!) \neq (\!|\, \langle r',\, q\rangle \bullet \mathrm{snd}[e'] \,|\!)$ *and*
$(\!|\, r \bullet e \,|\!) \neq (\!|\, \langle r'\rangle\mathrm{inl} \bullet [e',\, f] \,|\!)$ *and* $(\!|\, r \bullet e \,|\!) \neq (\!|\, \langle r'\rangle\mathrm{inr} \bullet [e',\, f] \,|\!)$

$(*14_R)\ (\!|\, \mu\alpha.\, c \bullet \underline{e} \,|\!)^* \equiv c^*\{\underline{e}^*/\alpha\} \qquad (*15_R)\ (\!|\, r \bullet \widetilde{\mu}x.\, c \,|\!)^* \equiv c^*\{r^*/x\}$

$(*16_R)\ (\!|\, \langle r,\, q\rangle \bullet \mathrm{fst}[e] \,|\!)^* \equiv (\!|\, r^* \bullet e^* \,|\!) \qquad (*17_R)\ (\!|\, \langle r,\, q\rangle \bullet \mathrm{snd}[e] \,|\!)^* \equiv (\!|\, q^* \bullet e^* \,|\!)$

$(*18_R)\ (\!|\, \langle r\rangle\mathrm{inl} \bullet [e,\, f] \,|\!)^* \equiv (\!|\, r^* \bullet e^* \,|\!) \qquad (*19_R)\ (\!|\, \langle r\rangle\mathrm{inr} \bullet [e,\, f] \,|\!)^* \equiv (\!|\, r^* \bullet f^* \,|\!)$

$$(*20_R)\ (\!|\, [e]\mathrm{not} \bullet \mathrm{not}\langle r\rangle \,|\!)^* \equiv (\!|\, r^* \bullet e^* \,|\!)$$

Theorem 7 (Star property for $\Rightarrow_{\mathrm{R}}$). *If $D \Rightarrow_R D'$ then $D' \Rightarrow_R D^*$.*

Now it is easy to deduce the diamond property for $\Rightarrow_{\mathrm{R}}$.

Theorem 8 (Diamond property for $\Rightarrow_{\mathrm{R}}$).
If $D_1 {}_R\!\Leftarrow D \Rightarrow_R D_2$ then $D_1 \Rightarrow_R D' {}_R\!\Leftarrow D_2$ for some D'.

Finally, from Lemma 5 and Theorem 8, it follows that $\mathsf{Dual}_{\mathrm{R}}$ is confluent.

Theorem 9 (Confluence of $\mathsf{Dual}_{\mathrm{R}}$).
If $D_1 \underset{R}{\twoheadleftarrow} D \underset{R}{\twoheadrightarrow} D_2$ then $D_1 \underset{R}{\twoheadrightarrow} D' \underset{R}{\twoheadleftarrow} D_2$ for some D'.

5 Type assignment system

A complementary perspective to that of considering the dual calculus as term-assignment to logic proofs is that of viewing sequent proofs as typing derivations for raw expressions. The set of types corresponds to the logical connectives; for the dual calculus the set of types is given by closing a set of *base types* X under conjunction, disjunction, and negation.

$$\mathsf{Type}:\ A,B\ ::=\ X \mid A \wedge B \mid A \vee B \mid \neg A$$

Type bases have two components, the *antecedent,* a set of bindings of the form $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, and the *succedent* of the form $\Delta = \alpha_1 : B_1, \ldots, \alpha_k : B_k$, where x_i, α_j are distinct for all $i = 1, \ldots, n$ and $j = 1, \ldots, k$.

The judgements of the type system are given by the following:

$$\Gamma \vdash \Delta, \boxed{r : A} \qquad \boxed{e : A}, \Gamma \vdash \Delta \qquad c\ :\ (\Gamma \vdash \Delta)$$

where Γ is the antecedent and Δ is the succedent. The first judgement is the typing for a term, the second is the typing for a coterm and the third one is the typing for a statement. The box denotes a distinguished output or input, i.e. a place where the computation will continue or where it happened before.

The type assignment system for the dual calculus, introduced by Wadler [30, 31], is given in Figure 3.

6 Strong normalization of typeable D-expressions

Definition 10. *A* pair *is given by two sets T and C with $T \subseteq \Lambda_R$ and $C \subseteq \Lambda_L$. If each of the components of a pair is non-empty we refer to it as a* non-trivial *pair.*

The pair (T, C) is a stable *pair if each of T and C is non-empty and for every $r \in T$ and every $e \in C$, the statement $(\!|\, r \bullet e \,|\!)$ is SN.*

For example, the pair $(Var_{\mathrm{R}}, Var_{\mathrm{L}})$ is stable. Note that the terms and coterms in any stable pair are themselves SN. Script letters will denote pairs. If $\mathcal{P}$ is a pair, $\mathcal{P}_{\mathrm{R}}$ and $\mathcal{P}_{\mathrm{L}}$ denote its component sets of terms and coterms.

We can use pairs to interpret types; the following technical condition will be crucial.

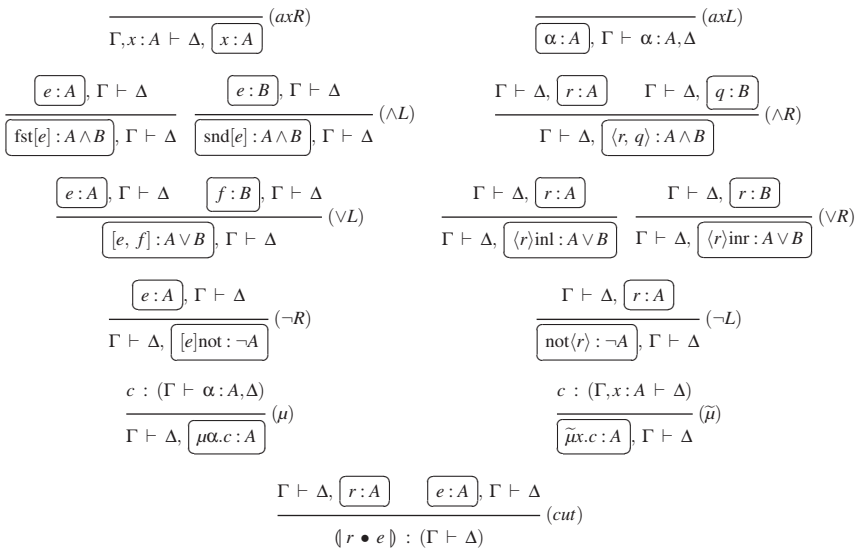

Fig. 3. Type system for the dual calculus

Definition 11. *A pair* (T,C) *is* saturated *if*

- T *contains all term variables and* C *contains all coterm variables,*
- *whenever* $\mu\alpha.c$ *satisfies* $\forall e \in C, c\{e/\alpha\}$ *is SN then* $\mu\alpha.c \in T$*, and*
- *whenever* $\tilde{\mu}x.c$ *satisfies* $\forall r \in T, c\{r/x\}$ *is SN then* $\tilde{\mu}x.c \in C$*.*

A pair (T,C) *is* simple *if no term in* T *is of the form* $\mu\alpha.c$ *and no coterm in* C *is of the form* $\tilde{\mu}x.c$*.*

We can always expand a pair to be saturated. The next result shows that if the original pair is stable and simple, then we may always arrange that the saturated extension is stable. The technique is similar to the "symmetric candidates" technique as used by Barbanera and Berardi [3] for the Symmetric Lambda Calculus and further adapted by Polonovski [25] in his proof of strong normalization for $\bar{\lambda}\mu\tilde{\mu}$ calculus with explicit substitutions.

Note that the saturation condition on variables is no obstacle to stability: it is easy to see that if (T,C) is any stable pair, then the pair obtained by adding all term variables to T and all coterm variables to C will still be stable.

Lemma 12. *Let* (T,C) *be a simple stable pair. Then there is an extension of* (T,C) *which is saturated and stable.*

Proof. As observed above, we may assume without loss of generality that T already contains all term variables and C already contains all coterm variables.

Define the maps $\widetilde{\Phi}_C : \Lambda_R \to \Lambda_L$ and $\Phi_T : \Lambda_L \to \Lambda_R$ by

$$\widetilde{\Phi}_C(T) = C \cup \{\widetilde{\mu}x.c \mid \forall r \in T, c\{r/x\} \text{ is SN}\}$$
$$\Phi_T(C) = T \cup \{\mu\alpha.c \mid \forall e \in C, c\{e/\alpha\} \text{ is SN}\}$$

Each of Φ_T and $\widetilde{\Phi}_C$ is antimonotone. So the map $\Phi_T \circ \widetilde{\Phi}_C : \Lambda_R \to \Lambda_R$ is monotone (indeed it is continuous).

Let T^* be any fixed point of $(\Phi_T \circ \widetilde{\Phi}_C)$; then take C^* to be $\widetilde{\Phi}_C(T^*)$. Since $T^* = \Phi_T(\widetilde{\Phi}_C(T^*))$ we have

$$T^* = \Phi_T(C^*) = T \cup \{\mu\alpha.c \mid \forall e \in C^*, c\{e/\alpha\} \text{ is SN}\} \quad \text{and} \tag{1}$$
$$C^* = \widetilde{\Phi}_C(T^*) = C \cup \{\widetilde{\mu}x.c \mid \forall r \in T^*, c\{r/x\} \text{ is SN}\} \tag{2}$$

It follows easily that $T \subseteq T^*$ and $C \subseteq C^*$ and that (T^*, C^*) is saturated. It remains to show that (T^*, C^*) is stable.

Since T is a set of SN terms and $C \neq \emptyset$, $\Phi_T(C)$ is a set of SN terms; similarly $\widetilde{\Phi}_C(T)$ is a set of SN coterms. The key fact is that, since (T,C) was simple, a term $\mu\alpha.c$ is in T^* *iff* $\forall e \in C^*, c\{e/\alpha\}$is SN: this is because a μ-term is in T^* precisely if it is in $\Phi_T(C^*) \setminus T$. Similarly a coterm $\widetilde{\mu}x.c$ is in C^* if and only if $\forall r \in T^*, c\{r/x\}$ is SN.

So consider any statement $(\!| r \bullet e |\!)$ with $r \in T^*$ and $e \in C^*$; we must show that this statement is SN. If in fact $r \in T$ and $e \in C$ then $(\!| r \bullet e |\!)$ is SN since (T,C) was stable.

So suppose $r \in (T^* \setminus T)$ and/or $e \in (C^* \setminus C)$, and consider any reduction sequence out of $(\!| r \bullet e |\!)$. If no top-level (μ- or $\widetilde{\mu}$-) reduction is ever done then the reduction must be finite since r and e are individually SN. If a top-level reduction is ever done then (cf Remark 1) we may promote this to be the first step, so that the reduction sequence begins $(\!| \mu\alpha.c \bullet e |\!) \longrightarrow c\{e/\alpha\}$ or $(\!| r \bullet \widetilde{\mu}x.c |\!) \longrightarrow c\{r/x\}$. But we observed above that in these cases the reduced D-expression is SN by definition of (T^*, C^*) and so our reduction is finite in length. □

6.1 Pairs and types

As a preliminary step in building pairs to interpret types we define the following constructions on pairs.

Definition 13. *Let $\mathcal{P}$ and $\mathcal{Q}$ be pairs.*

- *The pair $(\mathcal{P} \curlywedge \mathcal{Q})$ is given by:*
  - $(\mathcal{P} \curlywedge \mathcal{Q})_R = \{\langle r_1, r_2\rangle \mid r_1 \in \mathcal{P}_R, r_2 \in \mathcal{Q}_R\}$
  - $(\mathcal{P} \curlywedge \mathcal{Q})_L = \{\mathrm{fst}[e] \mid e \in \mathcal{P}_L\} \cup \{\mathrm{snd}[e] \mid e \in \mathcal{Q}_L\}$.
- *The pair $(\mathcal{P} \curlyvee \mathcal{Q})$ is given by:*
  - $(\mathcal{P} \curlyvee \mathcal{Q})_R = \{\langle r\rangle\mathrm{inl} \mid r \in \mathcal{P}_R\} \cup \{\langle r\rangle\mathrm{inr} \mid r \in \mathcal{Q}_R\}$.
  - $(\mathcal{P} \curlyvee \mathcal{Q})_L = \{[e_1, e_2] \mid e_1 \in \mathcal{P}_L.\ e_2 \in \mathcal{Q}_L\}$
- *The pair $\mathcal{P}^\circ$ is given by:*
  - $(\mathcal{P}^\circ)_R = \{[e]\mathrm{not} \mid e \in \mathcal{P}_L\}$
  - $(\mathcal{P}^\circ)_L = \{\mathrm{not}\langle r\rangle \mid r \in \mathcal{P}_R\}$

Note that each of $(\mathcal{P} \curlywedge \mathcal{Q})$, $(\mathcal{P} \curlyvee \mathcal{Q})$, and $\mathcal{P}^\circ$ is simple.

Lemma 14. *Let $\mathcal{P}$ and Q be stable pairs. Then $(\mathcal{P} \curlywedge Q)$, $(\mathcal{P} \curlyvee Q)$, and $\mathcal{P}^\circ$ are each stable.*

Proof. For $(\mathcal{P} \curlywedge Q)$: Let $r \in (\mathcal{P} \curlywedge Q)_{\mathrm{R}}$ and $e \in (\mathcal{P} \curlywedge Q)_{\mathrm{L}}$. We need to show that $(\!| r \bullet e |\!)$ is SN. Since $\mathcal{P}$ and Q are stable, it is easy to see that each of r and e is SN. So to complete the argument it suffices to show, again by the fact that top-level reductions can be promoted to be the first step in a reduction sequence, that the result of a top-level reduction is SN. Consider, without loss of generality, $(\!| \langle r_1, r_2 \rangle \bullet \mathrm{fst}[e] |\!) \longrightarrow (\!| r_1 \bullet e |\!)$. Then $r_1 \in \mathcal{P}_{\mathrm{R}}$ and $e \in \mathcal{P}_{\mathrm{L}}$, and since $\mathcal{P}$ is stable $(\!| r_1 \bullet e |\!)$ is SN, as desired.

The arguments for $(\mathcal{P} \curlyvee Q)$ and $\mathcal{P}^\circ$ are similar. □

The following is our notion of reducibility candidates for the dual calculus.

Definition 15. *The type-indexed family of pairs $\mathcal{S} = \{\mathcal{S}^T \mid T \text{ a type}\}$ is defined as follows.*

- *When T is a base type, $\mathcal{S}^T$ is any stable saturated extension of (Var_R, Var_L).*
- *$\mathcal{S}^{A \wedge B}$ is any stable saturated extension of $(\mathcal{S}^A \curlywedge \mathcal{S}^B)$.*
- *$\mathcal{S}^{A \vee B}$ is any stable saturated extension of $(\mathcal{S}^A \curlyvee \mathcal{S}^B)$.*
- *$\mathcal{S}^{\neg A}$ is any stable saturated extension of $(\mathcal{S}^A)^\circ$.*

The construction of each pair $\mathcal{S}^T$ succeeds by Lemma 12 and Lemma 14. Note that by definition of saturation each $\mathcal{S}^T$ contains all term variables and all coterm variables.

6.2 Strong normalization

Strong normalization of our calculus will follow if we establish the fact that typeable terms and coterms lie in the candidates $\mathcal{S}$.

Theorem 16. *If term r is typeable with type A then r is in $\mathcal{S}^A_R$; if coterm e is typeable with type A then e is in $\mathcal{S}^A_L$.*

Proof. To prove the theorem it is convenient, as usual, to prove a stronger statement. Say that a substitution θ *satisfies* Γ if

$$\forall (x : A) \in \Gamma, \;\; \theta x \in \mathcal{S}^A_{\mathrm{R}},$$

and that θ *satisfies* Δ if

$$\forall (\alpha : A) \in \Delta, \;\; \theta\alpha \in \mathcal{S}^A_{\mathrm{L}}.$$

Then the theorem follows from the assertion

suppose that θ satisfies Γ and Δ.
- *If $\Gamma \vdash \boxed{r : A}, \Delta$, then $\theta r \in \mathcal{S}^A_R$.*
- *If $\Gamma, \boxed{e : A} \vdash \Delta$, then $\theta e \in \mathcal{S}^A_L$.*

since the identity substitution satisfies every Γ and Δ.

Choose a substitution θ which satisfies Γ and Δ, and a typeable term r or a coterm e; we wish to show that $\theta r \in \mathcal{S}^T_{\mathrm{R}}$ or $\theta e \in \mathcal{S}^T_{\mathrm{L}}$, as appropriate. We prove the statement above by induction on typing derivations, considering the possible forms of the typing in turn. For lack of space we only show a representative sample of cases here.

Case: When the derivation consists of an axiom the result is immediate since θ satisfies Γ and Δ.

Case: Suppose the derivation ends with rule $(\wedge L)$. Without loss of generality we examine fst[ ]:

$$\dfrac{\boxed{e : A},\ \Gamma \vdash \Delta}{\boxed{\mathrm{fst}[e] : A \wedge B},\ \Gamma \vdash \Delta}$$

We wish to show that $\theta\mathrm{fst}[e] \equiv \mathrm{fst}[\theta e] \in \mathcal{S}_{\mathrm{L}}^{A \wedge B}$. By induction hypothesis $\theta e \in \mathcal{S}_{\mathrm{L}}^{A}$ and so $\mathrm{fst}[\theta e] \in (\mathcal{S}^{\mathcal{A}} \curlywedge \mathcal{S}^{\mathcal{B}})_{\mathrm{L}} \subseteq \mathcal{S}_{\mathrm{L}}^{A \wedge B}$.

Case: Suppose the derivation ends with rule $(\wedge R)$.

$$\dfrac{\Gamma \vdash \Delta, \boxed{r : A} \qquad \Gamma \vdash \Delta, \boxed{q : B}}{\Gamma \vdash \Delta, \boxed{\langle r, q \rangle : A \wedge B}}\ (\wedge R)$$

We wish to show that $\theta\langle r, q\rangle \equiv \langle \theta r, \theta q\rangle \in \mathcal{S}_{\mathrm{R}}^{A \wedge B}$. By induction hypothesis $\theta r \in \mathcal{S}_{\mathrm{R}}^{A}$ and $\theta q \in \mathcal{S}_{\mathrm{R}}^{B}$, and so $\langle \theta r, \theta q\rangle \in (\mathcal{S}^{\mathcal{A}} \curlywedge \mathcal{S}^{\mathcal{B}})_{\mathrm{R}} \subseteq \mathcal{S}_{\mathrm{R}}^{A \wedge B}$.

Case: Suppose the derivation ends with rule $(\neg L)$.

$$\dfrac{\Gamma \vdash \Delta, \boxed{r : A}}{\boxed{\mathrm{not}\langle r\rangle : \neg A},\ \Gamma \vdash \Delta}\ (\neg L)$$

We wish to show that $\theta\mathrm{not}\langle e\rangle \equiv \mathrm{not}\langle \theta e\rangle \in \mathcal{S}_{\mathrm{L}}^{\neg A}$. By induction hypothesis $\theta e \in \mathcal{S}_{\mathrm{L}}^{A}$, and so $\mathrm{not}\langle \theta e\rangle \in (\mathcal{S}^{\mathcal{A}^{\circ}})_{\mathrm{L}} \subseteq \mathcal{S}_{\mathrm{R}}^{\neg A}$.

Case: Suppose the derivation ends with rule (μ).

$$\dfrac{\dfrac{\Gamma \vdash \boxed{r : T}, \alpha : A, \Delta \quad \Gamma, \boxed{e : T} \vdash \alpha : A, \Delta}{(\!|\, r \bullet e \,|\!) : (\Gamma \vdash \alpha : A, \Delta)}\ (cut)}{\Gamma \vdash \boxed{\mu\alpha.(\!|\, r \bullet e \,|\!) : A}, \Delta}\ (\mu)$$

Note that any application of the typing rule (μ) must indeed immediately follow a cut. We wish to show that $\mu\alpha.(\!|\, \theta r \bullet \theta e \,|\!) \in \mathcal{S}_{\mathrm{R}}^{A}$.

Since $\mathcal{S}^{A}$ is saturated, to show this it suffices to show that for each $e_1 \in \mathcal{S}_{\mathrm{L}}^{A}$

$$(\!|\, \theta r \bullet \theta e \,|\!)\{e_1/\alpha\} \quad \text{is SN.}$$

Letting θ' denote the substitution obtained by augmenting θ with the binding $\alpha \mapsto e_1$, what we want to show is that $(\!|\, \theta' r \bullet \theta' e \,|\!)$ is SN.

The substitution θ' satisfies the basis $\alpha : A, \Delta$ by hypothesis and the fact that $e_1 \in \mathcal{S}_{\mathrm{L}}^{A}$. So $\theta' r \in \mathcal{S}_{\mathrm{R}}^{T}$ and $\theta' e \in \mathcal{S}_{\mathrm{L}}^{T}$ by induction hypothesis, so $(\!|\, \theta' r \bullet \theta' e \,|\!)$ is SN.

Case: When the derivation ends with rule $(\widetilde{\mu})$ the argument is similar to the (μ) case.

The remaining cases are each similar to one of those above. □

Theorem 17. *Every typeable term, coterm, and statement is SN.*

Proof. If t is a term [respectively, e is a coterm] typeable with type A then by Theorem 16 we have $t \in \mathcal{S}_{\mathrm{R}}^{A}$ [respectively, $\mathcal{S}_{\mathrm{L}}^{A}$], and each of these consists of SN expressions. If $t = c$ is a typeable statement then it suffices to observe that, taking α to be any covariable not occurring in c, the term $\mu\alpha.c$ is typeable. □

6.3 Extensionality and expansion rules

The equations of the dual calculus of [31] include a group of equations called "η-equations" which express extensionality properties. A typical equation for a term of type $A \wedge B$ is

$$(\eta\wedge) \qquad r = \langle \mu\alpha.(\!|\, r \bullet \mathrm{fst}[\alpha] \,|\!),\ \mu\beta.(\!|\, r \bullet \mathrm{snd}[\beta] \,|\!) \rangle$$

and there are similar equations for the other types. In traditional λ-calculus it has been found convenient to orient such equations from left to right, i.e. as expansions, as a tool for analyzing the equality relation.

As with all expansions there are obvious situations which allow immediate infinite application of the rules (see for example [13] or [6] for a discussion in the setting of the lambda-calculus). For example, we must forbid application of the above expansion rule to a term already of the form $\langle r_1,\, r_2 \rangle$ to prevent an infinite reduction. Slightly more subtly, if the term r is already part of a statement whose other side is one of the forms $\mathrm{fst}[e]$ or $\mathrm{snd}[e]$ then we can immediately fall into a cycle of $(\eta\wedge);(\beta\wedge)$ reductions.

But if we forbid only those clearly ill-advised situations, the result is a reduction relation with all the nice properties one might want. Lack of space forbids a detailed treatment here but the key points are as follows.

- The constraints on the expansion relation do not change the equalities we can prove, even under restrictions such as call-by-name or call-by-value, in the sense that if a term t can be expanded to term t' by a "forbidden" expansion, then t' can be *reduced* to t by one of the "computational" reductions (i.e., those from Figure 1).
- The resulting reduction relation is SN on typed terms.

It is straightforward to verify the first assertion. The second claim is proved by precisely the same techniques presented in the current section: the notions of saturated stable pair is robust enough so that there are no conceptual difficulties in accommodating expansions. Details will appear in the full version of the paper.

7 Conclusion

We have explored some aspects of the reduction relation on raw expressions of the dual calculus, and proven strong normalization and confluence results for several variations on the basic system.

An interesting open problem is to find a *characterization* of the SN terms, presumably in the form of an extension of the system of simple types studied here. For traditional λ-calculus, system of intersection types have been an invaluable tool in studying reduction properties, characterizing strong-, weak- and head-normalization. As shown in [12], subtle technical problems arise with the interaction between intersection types and symmetric calculi, so this promises to be a challenging line of inquiry.

References

1. Z. M. Ariola and H. Herbelin. Minimal classical logic and control operators. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, volume 2719 of *LNCS*, pages 871–885. sv, 2003.
2. S. v. Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In *ICTCS 2005 Ninth Italian Conference on Theoretical Computer Science, Certosa di Pontignano (Sienna), Italy*, 2005.
3. F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation*, 125(2):103–117, 1996.
4. H. P. Barendregt. *The Lambda Calculus: its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.
5. G. M. Bierman. A computational interpretation of the $\lambda\mu$-calculus. In *Proc. of Symposium on Mathematical Foundations of Computer Science.*, volume 1450 of *LNCS*, pages 336–345. Springer-Verlag, 1998.
6. R. D. Cosmo and D. Kesner. Simulating expansions without expansions. *Mathematical Structures in Computer Science*, 4(3):315–362, 1994.
7. P.-L. Curien. Symmetry and interactivity in programming. *Archive for Mathematical Logic*, 2001. to appear.
8. P.-L. Curien. Abstract machines, control, and sequents. In *Applied Semantics, International Summer School, APPSEM 2000, Advanced Lectures*, volume 2395 of *LNCS*, pages 123–136. Springer-Verlag, 2002.
9. P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of the 5th ACM SIGPLAN Int. Conference on Functional Programming (ICFP'00)*, Montreal, Canada, 2000. ACM Press.
10. R. David and K. Nour. Arithmetical proofs of strong normalization results for the symmetric $\lambda\mu$-calculus. In *TLCA*, pages 162–178, 2005.
11. P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In Springer-Verlag, editor, *LPAR'94*, volume 822 of *LNCS*, pages 31–43, 1994.
12. D. Dougherty, S. Ghilezan, and P. Lescanne. Characterizing strong normalization in a language with control operators. In *Sixth ACM SIGPLAN Conference on Principles and Practice of Declarative Programming PPDP'04*, pages 155–166. ACM Press, 2004.
13. D. J. Dougherty. Some lambda calculi with categorical sums and products. In C. Kirchner, editor, *Proc. 5th International Conference on Rewriting Techniques and Applications (RTA)*, volume 690 of *LNCS*, pages 137–151, Berlin, 1993. Springer-Verlag.
14. A. Filinski. Declarative continuations and categorical duality. Master's thesis, DIKU, Computer Science Department, University of Copenhagen, Aug. 1989. DIKU Rapport 89/11.
15. G. Gentzen. Unterschungen über das logische Schliessen, Math Z. 39 (1935), 176–210. In M. Szabo, editor, *Collected papers of Gerhard Gentzen*, pages 68–131. North-Holland, 1969.
16. T. Griffin. A formulae-as-types notion of control. In *POPL 17*, pages 47–58, 1990.
17. H. Herbelin. *Séquents qu'on calcule : de l'interprétation du calcul des séquents comme calcul de λ-termes et comme calcul de stratégies gagnantes*. Thèse, U. Paris 7, Janvier 1995.

18. W. A. Howard. The formulas-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490, New York, 1980. Academic Press.
19. S. Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. In B. Gramlich and S. Lucas, editors, *ENTCS*, volume 86. Elsevier, 2003.
20. S. Likavec. *Types for object oriented and functional programming languages*. PhD thesis, Università di Torino, Italy, ENS Lyon, France, 2005.
21. C. R. Murthy. Classical proofs as programs: How, what, and why. In J. P. M. Jr. and M. J. O'Donnell, editors, *Constructivity in Computer Science*, volume 613 of *LNCS*, pages 71–88. Springer, 1991.
22. C.-H. L. Ong and C. A. Stewart. A Curry-Howard foundation for functional computation with control. In *POPL 24*, pages 215–227, 1997.
23. M. Parigot. An algorithmic interpretation of classical natural deduction. In *Proc. of Int. Conf. on Logic Programming and Automated Reasoning, LPAR'92*, volume 624 of *LNCS*, pages 190–201. Springer-Verlag, 1992.
24. M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *The J. of Symbolic Logic*, 62(4):1461–1479, December 1997.
25. E. Polonovski. Strong normalization of $\lambda\mu\tilde{\mu}$-calculus with explicit substitutions. In I. Walukiewicz, editor, *Foundations of Software Science and Computation Structures, 7th International Conference, FOSSACS 2004*, volume 2987 of *LNCS*, pages 423–437. Springer, 2004.
26. D. Pym and E. Ritter. On the semantics of classical disjunction. *J. of Pure and Applied Algebra*, 159:315–338, 2001.
27. P. Selinger. Control categories and duality: On the categorical semantics of the lambda-mu calculus. *Mathematical Structures in Computer Science*, 11(2):207–260, 2001.
28. M. Takahashi. Parallel reduction in λ-calculus. *Information and Computation*, 118:120–127, 1995.
29. C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. In *Typed Lambda Calculus and Applications*, volume 1581 of *LNCS*, pages 365–380, 1999.
30. P. Wadler. Call-by-value is dual to call-by-name. In *Proc. of the 8th Int. Conference on Functional Programming*, pages 189–201, 2003.
31. P. Wadler. Call-by-value is dual to call-by-name, reloaded. In *Rewriting Technics and Application, RTA'05*, volume 3467 of *LNCS*, pages 185–203, 2005.