Classical logic with Mendler induction

MARCO DEVESAS CAMPOS, *Department of Computer Science and Technology, University of Cambridge, CB3 0FD Cambridge, United Kingdom.*
E-mail: devesas.campos@gmail.com

MARCELO FIORE, *Department of Computer Science and Technology, University of Cambridge, CB3 0FD Cambridge, United Kingdom.*

Abstract

We investigate (co-) induction in classical logic under the propositions-as-types paradigm, considering propositional, second-order and (co-) inductive types. Specifically, we introduce an extension of the Dual Calculus with a Mendler-style (co-) iterator and show that it is strongly normalizing. We prove this using a reducibility argument.

1 Introduction

The Curry–Howard isomorphism. The interplay between logic and computer science has a long and rich history. In particular, the Curry–Howard isomorphism, the correspondence between types and propositions, and between typings and proofs, is a long established bridge through which results in one field can fruitfully migrate to the other.

Classical calculi. One such example—motivating of the research presented herein—is the use of programming calculi based on Gentzen's sequent calculus *LK* [11]. At its core, *LK* is a calculus of the dual concepts of necessary assumptions and possible conclusions—concepts that map neatly, on the computer science side, to required inputs (or computations) and possible outputs (or continuations). As an example of the kind of analysis that can be done by focusing on this separation of concepts, Curien and Herbelin [6] and Wadler [22] devised *LK*-based calculi that showed, syntacticly, the duality of the two most common evaluations strategies: call-by-name and call-by-value. While originally such classical calculi included only propositional types—i.e. conjunction, disjunction, negation, implication and subtraction (the dual connective of implication)—they were later extended with second-order types [15, 20] and also with *positive* (co-) inductive types [15]. The latter fundamentally depended on the existence of a map operation of the underlying type. Here we do away with this restriction.

Mendler induction. We take as primitive a more general induction scheme based on the work of Mendler [17]. He proposed an induction operator for System F that operated in a way akin to the call-by-name fix-point combinator—thus avoiding direct use of said mapping operations—but ensured termination by clever use of polymorphic types. Further adding to its generality, it was later shown that with Mendler's iterator one could in fact induct on data-types of *arbitrary variance*—i.e. data-types whose induction variable may also appear negatively [16, 21]. Due to its generality, Mendler induction has been applied in a number of different contexts, such as higher-order recursive types [1, 2] and automated theorem proving [13].

Classical logic *and* Mendler induction. The central question of this paper is whether Mendler induction can be lifted to non-functional settings without introducing ill-effects—specifically, non-

Vol. 00, No. 0,

doi:10.1093/logcom/exaa004

TABLE 1 Syntax of the second-order Dual Calculus.

Terms

$$t := x, y, \ldots \in \mathit{Var} \mid \alpha.(c)$$
$$\mid \underbrace{\langle t, t' \rangle \mid i_1 \langle t \rangle \mid i_2 \langle t \rangle \mid \mathit{not}\langle k \rangle \mid \lambda x.\,(t) \mid (t\#k) \mid a\,\langle t \rangle \mid e\,\langle t \rangle}_{\text{Introductions}}$$

Co-terms

$$k := \alpha, \beta, \ldots \in \mathit{Covar} \mid x.\,(c)$$
$$\mid \underbrace{[k, k'] \mid \mathit{fst}[k] \mid \mathit{snd}\,[k] \mid \mathit{not}[t] \mid (t@k) \mid \mu\alpha.(k) \mid a\,[k] \mid e\,[k]}_{\text{Eliminations}}$$

Cuts

$$c := t \bullet k$$

termination. Logically, this entails the consistency of a classical system that goes further than the usual Boolean and second-order propositions. There is no *a priori* reason to assume this might be the case: languages based on classical logic have been shown to be quite misbehaved if not handled properly [12] and certain forms of Mendler induction have been shown to break strong normalization at higher-ranked types [2].

Here we show that these two constructions are, in fact, compatible. In summary, we:

- develop a second-order Dual Calculus with functional types (Section 2),
- prove its strong normalization (Section 3) via a reducibility argument,
- review Mendler induction in a functional setting (Section 4),
- extend the Dual Calculus with Mendler (co-) inductive types (Section 5) and
- adapt the aforementioned reducibility argument to prove that the extension is also strongly normalizing (Section 6).

Duality. At every stage—borrowing from one of *LK*'s design principles—we consider concomitantly the duals of every type we introduce, viz. subtraction [5] and co-induction. Similarly to *LK*, this entails little more than 'flipping' the actions on the left and on the right. The choice to do so was not merely aesthetic: having subtractions in our system affords us a much more natural definition of Mendler induction than if we only had implication at our disposal. This is comparable to the use of existential types as a basis for modeling ad hoc polymorphism in functional languages [18] as opposed to the more elaborate encoding by means of universal types—and stresses the point that duality brings forth gains in expressiveness at little cost for the designer.

2 Second-order Dual Calculus

The base calculus. Our base formalism is Wadler's Dual Calculus [22]—often abbreviated DC We begin by reviewing the original propositional version extended with second-order types [15] and subtractive types [5, 6]. Tables 1[1],2 and 3 respectively summarize the syntax, the types and typing rules, and the reduction rules of the calculus.

Syntax. The sequent calculus *LK* is a calculus of multiple assumptions and conclusions, as witnessed

[1]Unlike Wadler's presentation, we keep the standard practice of avoiding suffix operators—while lexical duality is lost, we argue that it improves readability.

TABLE 2 Typing for the second-order propositional Dual Calculus (with the structural rules omitted).

Types

$$T, A, B := X \mid A \wedge B \mid A \vee B \mid \neg A \mid A \to B \mid A - B \mid \forall X\,.\,T \mid \exists X\,.\,T$$

Identity

$$\frac{}{x{:}A \vdash x{:}A \mid} \qquad \frac{}{\mid \alpha{:}A \dashv \alpha{:}A}$$

Abstractions

$$\frac{\Gamma \vdash c \dashv \Delta, \alpha{:}A}{\Gamma \vdash \alpha.(c){:}A \mid \Delta} \qquad \frac{x{:}A, \Gamma \vdash c \dashv \Delta}{\Gamma \mid x.(c){:}A \dashv \Delta}$$

Cut

$$\frac{\Gamma \vdash t{:}A \mid \Delta \qquad \Gamma \mid k{:}A \dashv \Delta}{\Gamma \vdash t \bullet k \dashv \Delta}$$

Conjunction

$$\frac{\Gamma \vdash t{:}A \mid \Delta \qquad \Gamma \vdash t'{:}B \mid \Delta}{\Gamma \vdash \langle t, t' \rangle{:}A \wedge B \mid \Delta}$$

$$\frac{\Gamma \mid k{:}A \dashv \Delta}{\Gamma \mid fst[k]{:}A \wedge B \dashv \Delta} \qquad \frac{\Gamma \mid k{:}B \dashv \Delta}{\Gamma \mid snd[k]{:}A \wedge B \dashv \Delta}$$

Disjunction

$$\frac{\Gamma \vdash t{:}A \mid \Delta}{\Gamma \vdash i_1\langle t \rangle{:}A \vee B \mid \Delta} \qquad \frac{\Gamma \vdash t{:}B \mid \Delta}{\Gamma \vdash i_2\langle t \rangle{:}A \vee B \mid \Delta}$$

$$\frac{\Gamma \mid k{:}A \dashv \Delta \qquad \Gamma \mid k'{:}B \dashv \Delta}{\Gamma \mid [k, k']{:}A \vee B \dashv \Delta}$$

Negation

$$\frac{\Gamma \mid k{:}A \dashv \Delta}{\Gamma \vdash not\langle k \rangle{:}\neg A \mid \Delta} \qquad \frac{\Gamma \vdash t{:}A \mid \Delta}{\Gamma \mid not[t]{:}\neg A \dashv \Delta}$$

Implication

$$\frac{x{:}A, \Gamma \vdash t{:}B \mid \Delta}{\Gamma \vdash \lambda x.(t){:}A \to B \mid \Delta} \qquad \frac{\Gamma \vdash t{:}A \mid \Delta \qquad \Gamma \mid k{:}B \dashv \Delta}{\Gamma \mid (t@k){:}A \to B \dashv \Delta}$$

Subtraction

$$\frac{\Gamma \vdash t{:}A \mid \Delta \qquad \Gamma \mid k{:}B \dashv \Delta}{\Gamma \vdash (t\#k){:}A - B \mid \Delta} \qquad \frac{\Gamma \mid k{:}A \dashv \Delta, \alpha{:}B}{\Gamma \mid \mu\alpha.(k){:}A - B \dashv \Delta}$$

Universal quantification

$$\frac{\Gamma \vdash t{:}F(X) \mid \Delta}{\Gamma \vdash a\langle t \rangle{:}\forall X.F(X) \mid \Delta} \ (X \text{ not free in } \Gamma, \Delta) \qquad \frac{\Gamma \mid k{:}F(A) \dashv \Delta}{\Gamma \mid a[k]{:}\forall X.F(X) \dashv \Delta}$$

Existential quantification

$$\frac{\Gamma \vdash t{:}F(A) \mid \Delta}{\Gamma \vdash e\langle t \rangle{:}\exists X.F(X) \mid \Delta} \qquad \frac{\Gamma \mid k{:}F(X) \dashv \Delta}{\Gamma \mid e[k]{:}\exists X.F(X) \dashv \Delta} \ (X \text{ not free in } \Gamma, \Delta)$$

by the action of the right and left derivation rules. Similarly, the two main components of DC are split into two kinds: *terms* (or *computations*) which, intuitively, produce values, and *co-terms* (or *continuations*), which consume them. However, whereas in the sequent calculus one can mix the different kinds of rules in any order, to keep the computational connection, the term and co-term

TABLE 3 Head reduction for the second-order Dual Calculus.

$\langle t, t' \rangle \bullet \mathit{fst}[k] \rightsquigarrow t \bullet k$	$\langle t, t' \rangle \bullet \mathit{snd}[k] \rightsquigarrow t' \bullet k$
$i_1 \langle t \rangle \bullet [k, k'] \rightsquigarrow t \bullet k$	$i_2 \langle t \rangle \bullet [k, k'] \rightsquigarrow t \bullet k'$
$\mathit{not}\langle k \rangle \bullet \mathit{not}[t] \rightsquigarrow t \bullet k$	
$\lambda x. (t) \bullet (t'@k) \rightsquigarrow t[t'/x] \bullet k$	$(t\#k) \bullet \mu\alpha.(k') \rightsquigarrow t \bullet k' [k/\alpha]$
$a \langle t \rangle \bullet a[k] \rightsquigarrow t \bullet k$	$e \langle t \rangle \bullet e[k] \rightsquigarrow t \bullet k$
$\alpha.(c) \bullet k \rightsquigarrow c[k/\alpha]$	$t \bullet x.(c) \rightsquigarrow c[t/x]$

formation rules are restricted in what phrases they expect—e.g. pairs should combine values, while projections pass the components of a pair to some other continuation. This distinction also forces the existence of two kinds of variables: variables for terms and co-variables for co-terms; we assume that they belong to some disjoint and countably infinite sets denoted by *Var* and *Covar*, respectively.

Cuts and abstractions. The third and final kind of phrase in the Dual Calculus are *cuts*. Recall the famous dictum of computer science:

$$\text{Data-structures} + \text{Algorithms} = \text{Programs}.$$

In DC, where terms represent the creation of information and co-terms consume it, we find that cuts, the combination of a term with a continuation, are analogous to programs:

$$\text{Terms} + \text{Co-terms} = \text{Cuts};$$

they are the entities that are capable of being executed. Given a cut, one can consider the execution that would ensue if given data for a variable or co-variable. The calculus provides a mechanism to express such situations by means of *abstractions* $x.(c)$ and of *co-abstractions* $\alpha.(c)$ on any cut c. Abstractions are continuations—they expect values in order to proceed with some execution—and, dually, co-abstractions are computations.

Subtraction. One novelty of this paper is the central rôle given to subtractive types, $A - B$ [5]. Subtraction is the dual connective to implication; it is to continuations what implication is to terms: it allows one to abstract co-variables in co-terms—and thereby *compose continuations*. Given a continuation k where a co-variable α might appear free, the subtractive abstraction (or catch, due to its connection with exception handling) is defined by the binding operator $\mu\alpha.(k)$—the idea being that applying (read, cutting) a continuation k' and value t to it, *packed* together as $(t\#k')$, yields a cut of the form $t \bullet k[k'/\alpha]$.

Typing judgments. We present the types and the typing rules in Table 2; we omit the structural rules here but they can be found in the aforementioned paper by Wadler [22]. We have three forms of typing judgments that go hand-in-hand with the three different types of phrases: $\Gamma \vdash t : A \mid \Delta$ for terms, $\Gamma \mid k : A \dashv \Delta$ for co-terms and $\Gamma \vdash c \dashv \Delta$ for cuts. The entailment symbol(s) always points to the phrase under judgment, and it appears in the same position as the entailment symbol in the logically corresponding sequent of *LK*. *Typing contexts* Γ are finite assignments of variables to their assumed types; dually, *typing co-contexts* Δ assign co-variables to their types. Tacitly, we assume that they always include the free (co-) variables in the phrase under consideration. Type-schemes $F(X)$ are types in which a distinguished type variable X may appear free; the instantiation of such a type-scheme to a particular type T is simply the (capture avoiding) substitution of the distinguished X by T and is denoted $F(T)$.

TABLE 4 Parallel reduction for the dual calculus.

$$c_0 \rightsquigarrow c_1 \implies c_0 \overset{\rightsquigarrow}{\rightsquigarrow} c_1$$

$$t_0 \overset{\rightsquigarrow}{\rightsquigarrow} t_1 \implies \begin{cases} \langle t_0, t\rangle \overset{\rightsquigarrow}{\rightsquigarrow} \langle t_1, t\rangle \\ \langle t, t_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} \langle t, t_1\rangle \\ i_1\langle t_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} i_1\langle t_1\rangle \\ i_2\langle t_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} i_2\langle t_1\rangle \\ \lambda x.(t_0) \overset{\rightsquigarrow}{\rightsquigarrow} \lambda x.(t_1) \\ (t_0 \# k) \overset{\rightsquigarrow}{\rightsquigarrow} (t_1 \# k) \\ a\langle t_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} a\langle t_1\rangle \\ e\langle t_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} e\langle t_1\rangle \\ not[t_0] \overset{\rightsquigarrow}{\rightsquigarrow} not[t_1] \end{cases} \qquad k_0 \overset{\rightsquigarrow}{\rightsquigarrow} k_1 \implies \begin{cases} fst[k_0] \overset{\rightsquigarrow}{\rightsquigarrow} fst[k_1] \\ snd[k_0] \overset{\rightsquigarrow}{\rightsquigarrow} snd[k_1] \\ [k, k_0] \overset{\rightsquigarrow}{\rightsquigarrow} [k, k_1] \\ [k_0, k] \overset{\rightsquigarrow}{\rightsquigarrow} [k_1, k] \\ \mu\alpha.(k_0) \overset{\rightsquigarrow}{\rightsquigarrow} \mu\alpha.(k_1) \\ (t \# k_0) \overset{\rightsquigarrow}{\rightsquigarrow} (t \# k_1) \\ a[k_0] \overset{\rightsquigarrow}{\rightsquigarrow} a[k_1] \\ e[k_0] \overset{\rightsquigarrow}{\rightsquigarrow} e[k_1] \\ not\langle k_0\rangle \overset{\rightsquigarrow}{\rightsquigarrow} not\langle k_1\rangle \end{cases}$$

$$c_0 \overset{\rightsquigarrow}{\rightsquigarrow} c_1 \implies \alpha.(c_0) \overset{\rightsquigarrow}{\rightsquigarrow} \alpha.(c_1) \qquad c_0 \overset{\rightsquigarrow}{\rightsquigarrow} c_1 \implies x.(c_0) \overset{\rightsquigarrow}{\rightsquigarrow} x.(c_1)$$

$$t_0 \overset{\rightsquigarrow}{\rightsquigarrow} t_1 \implies t_0 \bullet k \overset{\rightsquigarrow}{\rightsquigarrow} t_1 \bullet k \qquad k_0 \overset{\rightsquigarrow}{\rightsquigarrow} k_1 \implies t \bullet k_0 \overset{\rightsquigarrow}{\rightsquigarrow} t \bullet k_1$$

Example: witness the lack of witness. We can apply the rules in Table 2 to bear proof of valid formulas in second-order classical logic. One such example at the second-order level is $\neg\forall X\,.\,T \to \exists X\,.\,\neg T$:

$$\mid not[a\,\langle \alpha.(e\,\langle not\langle\alpha\rangle\rangle \bullet \beta)\rangle] : \neg\forall X\,.\,T \dashv \beta : \exists X\,.\,\neg T.$$

Note how the existential does not construct witnesses but simply diverts the flow of execution (by use of a co-abstraction).

Head reduction. The final ingredient of the calculus is the set of reduction rules. Head reduction rules (Table 3) encode the operational behavior that one would expect from the constructs of the language. They apply only to cuts and only at the outermost level. Head reduction is non-deterministic—as a cut made of abstractions and co-abstractions can reduce by either one of the abstraction rules—and non-confluent [22, p. 195]. Confluence can be reestablished by prioritizing the reduction of one type of abstraction over the other; this gives rise to two confluent reduction disciplines that we term *abstraction prioritizing* and *co-abstraction prioritizing*. In any case, reduction of well-typed cuts yields well-typed cuts.[2]

Parallel reduction. Since the phrases of DC are defined by mutual induction, we can generalize head-reduction to cuts that occur inside any term, co-term or, indeed, in other cuts (Table 4). Because, in general, several rules can be applied in parallel to any given phrase, we call this *parallel reduction*.

3 Strong normalization of the second-order Dual Calculus

The proof of strong normalization. Having surveyed the syntax, types and reduction rules of DC, we will now give a proof of its strong normalization—i.e. that all reduction sequences of well-typed phrases terminate in a finite number of steps—for the given *non-deterministic parallel reduction rules*. It follows that all manner of reduction sub-strategies, such as head reduction and the deterministic co- or abstraction prioritizing strategies, are also strongly normalizing.

[2]As we are not looking at call-by-name and call-by-value, we do not use the same reduction rule for implication as Wadler [22]; the rule here is due to Curien and Herbelin [6].

TABLE 5 Second-order type actions for orthogonal pairs.

$P \wedge Q$	$= \left(\left\langle (P)^{\mathsf{T}}, (Q)^{\mathsf{T}} \right\rangle, \mathit{fst}\left[(P)^{\mathsf{K}}\right] \cup \mathit{snd}\left[(Q)^{\mathsf{K}}\right]\right)$
$P \vee Q$	$= \left(i_1\left\langle (P)^{\mathsf{T}}\right\rangle \cup i_2\left\langle (Q)^{\mathsf{T}}\right\rangle, \left[(P)^{\mathsf{K}}, (Q)^{\mathsf{K}}\right]\right)$
$\neg P$	$= \left(\mathit{not}\left\langle (P)^{\mathsf{K}}\right\rangle, \mathit{not}\left[(P)^{\mathsf{T}}\right]\right)$
$P \to Q$	$= \bigvee_{x \in \mathit{Var}}\left(\lambda x.\left(\left((Q)^{\mathsf{T}} \vert^{(P)^{\mathsf{T}}/x}\right)\right), \left((P)^{\mathsf{T}} @ (Q)^{\mathsf{K}}\right)\right)$
$P - Q$	$= \bigwedge_{\alpha \in \mathit{Covar}}\left(\left((P)^{\mathsf{T}} \# (Q)^{\mathsf{K}}\right), \mu\alpha.\left((P)^{\mathsf{K}} \vert^{(Q)^{\mathsf{K}}/\alpha}\right)\right)$
$\forall S$	$= \bigwedge_{P \in S}\left(a\left\langle (P)^{\mathsf{T}}\right\rangle, a\left[(P)^{\mathsf{K}}\right]\right)$
$\exists S$	$= \bigvee_{P \in S}\left(e\left\langle (P)^{\mathsf{T}}\right\rangle, e\left[(P)^{\mathsf{K}}\right]\right)$

The proof rests on a reducibility argument. Similar approaches for the propositional fragment can be found in the literature [10, 20]; however, the biggest influence on our proof was the one by Parigot for the second-order extension of the Symmetric Lambda-Calculus [19].

Our main innovation here is the identification of a complete lattice structure with fix-points suitable for the interpretation of (co-) inductive types. We will, in fact, need to consider two lattices: $\mathcal{OP}$ and $\mathcal{ONP}$. Because types have structure in the form of terms and co-terms, each element of said lattices is a pair of sets, with terms in one component and co-terms in the other. These two must be *orthogonal*—i.e. all cuts formed with those terms and co-terms must be strongly normalizing. The difference between the two is that in $\mathcal{OP}$, we find terms/co-terms of arbitrary form; the components of lattice $\mathcal{ONP}$ are restricted to having only terms/co-terms that are introductions/eliminations. Between these two domains, we have type-induced *actions* from $\mathcal{OP}$ to $\mathcal{ONP}$ and a completion operator $\perp\!\!\perp$ from $\mathcal{ONP}$ to $\mathcal{OP}$ that generates all terms/co-terms compatible with the given introductions/eliminations.

$$\begin{tikzcd} \mathcal{OP} \arrow[r, bend left, "{\wedge,\vee,\neg,\ldots}"] & \mathcal{ONP} \arrow[l, bend left, "\perp\!\!\perp"] \end{tikzcd} \tag{1}$$

In this setting, we give (two) mutually induced interpretations for types (one in $\mathcal{ONP}$ and the other in $\mathcal{OP}$) and establish an adequacy result (Theorem 3.21) from which strong normalization follows as a corollary.

3.1 Operations on sets of syntax

Sets of syntax. The set of all terms formed using the rules in Table 1 will be denoted by $\mathcal{T}$; similarly, co-terms will be $\mathcal{K}$ and cuts $\mathcal{C}$. We will also need three special subsets of those sets: $\mathcal{IT}$ for those terms whose outer syntactic form is an introduction, $\mathcal{EK}$, dually, for the co-terms whose outer syntactic form is an eliminator and $\mathcal{SN}$ for the set of strongly normalizing cuts.[3] Since the proof

[3]A non-terminating, non-well-typed cut: $\alpha.(\mathit{not}\langle\alpha\rangle \bullet \alpha) \bullet \mathit{not}[\alpha.(\mathit{not}\langle\alpha\rangle \bullet \alpha)]$.

refers to the parallel reduction strategy, we also need the set of strongly normalizing terms, $\mathcal{SNT}$, and the set of strongly normalizing co-terms, $\mathcal{SNK}$.

Saturation. These sets above have all in common the property that they are closed under reduction. They are said to be saturated. For example, a strongly normalizing term must reduce to strongly normalizing terms and cuts reduce to cuts. Saturation can be expressed in terms of the image, denoted $[-]$, of the (parallel) reduction relation. In symbols, a set of phrases $P \in \mathcal{P}(\mathcal{T} \cup \mathcal{K} \cup \mathcal{C})$ is saturated *iff* $[\rightsquigarrow](P) \subseteq P$.

Syntactic actions on sets. The syntactic constructors give rise to obvious *actions* on sets of terms, co-terms and cuts, e.g.

$$- \bullet - : \mathcal{P}(\mathcal{T}) \times \mathcal{P}(\mathcal{K}) \to \mathcal{P}(\mathcal{C}), \quad T \bullet K = \{t \bullet k \mid t \in T, k \in K\}.$$

By abuse of notation, these operators shall be denoted as their syntactic counterparts. Apart from the cut action, which may introduce head reductions, they all preserve saturation.

LEMMA 3.1 (Saturation for (co-) term syntactic operators).
Assume T and U, K and L and C are saturated sets of terms, co-terms and cuts, respectively. Then the sets constructed from them using the introduction, elimination and structural-abstraction syntactic operators are all saturated:

$$\begin{array}{lll}
[\rightsquigarrow](\langle T, U\rangle) \subseteq \langle T, U\rangle & [\rightsquigarrow](\mathit{fst}[K]) \subseteq \mathit{fst}[K] & [\rightsquigarrow](\mathit{snd}[K]) \subseteq \mathit{snd}[K] \\
[\rightsquigarrow](i_1\langle T\rangle) \subseteq i_1\langle T\rangle & [\rightsquigarrow](i_2\langle T\rangle) \subseteq i_2\langle T\rangle & [\rightsquigarrow]([K, L]) \subseteq [K, L] \\
[\rightsquigarrow](\mathit{not}\langle K\rangle) \subseteq \mathit{not}\langle K\rangle & [\rightsquigarrow](\mathit{not}[T]) \subseteq \mathit{not}[T] & \\
[\rightsquigarrow](\lambda x.(T)) \subseteq \lambda x.(T) & [\rightsquigarrow]((T@K)) \subseteq (T@K) & \\
[\rightsquigarrow]((T\#K)) \subseteq (T\#K) & [\rightsquigarrow](\mu\alpha.(K)) \subseteq \mu\alpha.(K) & \\
[\rightsquigarrow](a\langle T\rangle) \subseteq a\langle T\rangle & [\rightsquigarrow](a[K]) \subseteq a[K] & \\
[\rightsquigarrow](e\langle T\rangle) \subseteq e\langle T\rangle & [\rightsquigarrow](e[K]) \subseteq e[K] & \\
[\rightsquigarrow](\alpha.(C)) \subseteq \alpha.(C) & [\rightsquigarrow](x.(C)) \subseteq x.(C) &
\end{array}$$

Substitution and its restriction. The (capture-avoiding) substitution operation lifts point-wise to the level of sets as a monotone function $(-)[(=)/\phi] : \mathcal{P}(U) \times \mathcal{P}(V) \to \mathcal{P}(U)$ for V the set of terms (resp. co-terms), ϕ a variable (resp. co-variable) and U either the set of terms, co-terms or cuts. We will often need to answer the question of which strongly normalizing phrases are in some set of 'good phrases' after simultaneously substituting for some variables. This is handled by the following operations of *restriction under (simultaneous, capture-avoinding) substitution*: given $\vec{\chi}$ a finite family of co-/variables and $\vec{P}$ a family of sets with each P_i in $\mathcal{T}$ or $\mathcal{K}$, as appropriate for the respective χ_i; the restriction to $T \in \mathcal{P}(\mathcal{T})$ under substitution $\left[\vec{P}/\vec{\chi}\right]$ is given by

$$T\Big|^{\vec{P}/\vec{\chi}} = \left\{t \in \mathcal{SNT} \,\middle|\, \forall \vec{\tau} \in \vec{P}\,.\, t[\vec{\tau}/\vec{\chi}] \in T\right\} \in \mathcal{P}(\mathcal{SNT});$$

the restrictions $-\Big|^{\vec{P}/\vec{\chi}} : \mathcal{P}(\mathcal{K}) \to \mathcal{P}(\mathcal{SNK})$ and $-\Big|^{\vec{P}/\vec{\chi}} : \mathcal{P}(\mathcal{C}) \to \mathcal{P}(\mathcal{SN})$ are defined similarly.

They are clearly antitone on $\vec{P}$ and monotone on the set we are restricting to. Furthermore, they witness an adjoint situation between sets of terms/co-terms/cuts and their strongly normalizing counterparts.

PROPOSITION 3.2
Let $\vec{\chi}$ be a finite family of variables and co-variables and $\vec{P}$ a(n equally indexed) family of sets of phrases, each according with the kind of the respective χ; then,

$$V\left[\vec{P}/\vec{\chi}\right] \subseteq_{\mathcal{P}(\mathcal{T})} T \text{ \textit{iff} } V \subseteq_{\mathcal{P}(\mathcal{SNT})} T\big|^{\vec{P}/\vec{\chi}}.$$

Substitutivity. The preservation of saturation by this restriction for one (co-) variable requires the concept *substitutivity* of reduction–substitution commutes with (parallel) reduction in the following two ways:

LEMMA 3.3
(I) For any phrases (i.e. terms/co-terms/cuts) $p_0 \overset{\sim}{\rightsquigarrow} p_1$

$$p_0 \overset{\sim}{\rightsquigarrow} p_1 \Longrightarrow \begin{cases} p_0\,[t/x] \overset{\sim}{\rightsquigarrow} p_1\,[t/x] \\ p_0\,[k/\alpha] \overset{\sim}{\rightsquigarrow} p_1\,[k/\alpha]\,. \end{cases}$$

(II) For any phrase p

$$t_0 \overset{\sim}{\rightsquigarrow} t_1 \Longrightarrow p\,[t_0/x] \overset{\sim}{\rightsquigarrow}^{*} p\,[t_1/x] \qquad k_0 \overset{\sim}{\rightsquigarrow} k_1 \Longrightarrow p\,[k_0/\alpha] \overset{\sim}{\rightsquigarrow}^{*} p\,[k_1/\alpha]\,.$$

The first property can take a more algebraic flavour:

PROPOSITION 3.4
Let P stand for a set of terms, or of co-terms or cuts, U be a set of terms and L a set of co-terms. Substitutivity I is equivalent to

$$([\overset{\sim}{\rightsquigarrow}](P))\,[U/x] \subseteq [\overset{\sim}{\rightsquigarrow}]\,(P\,[U/x]) \qquad ([\overset{\sim}{\rightsquigarrow}](P))\,[L/\alpha] \subseteq [\overset{\sim}{\rightsquigarrow}]\,(P\,[L/\alpha])\,.$$

Restriction and saturation. And all this algebraic scaffolding affords us a very straightforward proof of

THEOREM 3.5
Let P be a saturated set of either terms, co-terms or cuts; then, for any $T \subseteq \mathcal{T}$ and any $K \subseteq \mathcal{K}$, the sets $P\big|^{T/x}$ and $P\big|^{K/\alpha}$ are saturated.

PROOF. Note that $P\big|^{T/x} \subseteq \mathcal{SN}/\mathcal{SNT}/\mathcal{SNK}$ so, also, $[\overset{\sim}{\rightsquigarrow}](P\big|^{T/x}) \subseteq \mathcal{SN}/\mathcal{SNT}/\mathcal{SNK}$ (likewise for $P\big|^{K/\alpha}$). Then, for either case, a simple algebraic derivation—using the adjointness of the restrictions, monotonicity of the image of a relation, substitutivity and saturation—suffices.

$$\begin{array}{rrll}
 & P\big|^{T/x} & \subseteq P\big|^{T/x} & \\
\Longrightarrow & P\big|^{T/x}[T/x] & \subseteq P & (\textit{Prop. } 3.2) \\
\Longrightarrow & [\overset{\sim}{\rightsquigarrow}](P\big|^{T/x}[T/x]) & \subseteq [\overset{\sim}{\rightsquigarrow}]\,(P) & (\textit{Mon. of image}) \\
\Longrightarrow & ([\overset{\sim}{\rightsquigarrow}](P\big|^{T/x}))[T/x] & \subseteq [\overset{\sim}{\rightsquigarrow}]\,(P) & (\textit{Prop. } 3.4) \\
\Longrightarrow & ([\overset{\sim}{\rightsquigarrow}](P\big|^{T/x}))[T/x] & \subseteq P & (\textit{Saturation}) \\
\Longrightarrow & [\overset{\sim}{\rightsquigarrow}](P\big|^{T/x}) & \subseteq P\big|^{T/x} & (\textit{Prop. } 3.2).
\end{array}$$

□

3.2 *Orthogonal pairs*

Orthogonal pairs. Whenever a term t and a co-term k—necessarily strongly normalizing—form a strongly normalizing cut $t \bullet k$, we say that they are *orthogonal*. Similarly, for sets T of terms and

K of co-terms, we say that they are orthogonal if $T \bullet K \subseteq \mathcal{SN}$. We use the name *orthogonal pairs* for pairs of orthogonal sets that are saturated and denote the set of all such pairs by $\mathcal{OP}$. For any orthogonal pair $P \in \mathcal{OP}$, its set of terms is denoted $(P)^{\mathsf{T}}$ and its set of co-terms by $(P)^{\mathsf{K}}$. Note that no type restriction is in play in the definition of orthogonal pairs, e.g. a cut of an injection with a projection is by definition orthogonal as no reduction rule applies.

Lattices. Recall that a lattice L is a partially ordered set such that every *non-empty* finite subset S of the carrier of L has a least upper bound (or join, or lub) and a greatest lower-bound (or meet, or glb), respectively, denoted by $\bigvee S$ and $\bigwedge S$. By abuse of notation, we conflate lattices with their carrier sets unless otherwise noted. If the bounds exist for any subset of L, one says that the lattice is *complete*. In particular, this entails the existence of a bottom and a top element for the partial order. The powerset $\mathcal{P}(S)$ of a set S ordered by inclusion, together with set-union and set-intersection, is a complete lattice with bounds given by S and the empty set. The dual L^{op} of a (complete) lattice L (where we take the opposite order and invert the bounds) is a (complete) lattice, as is the point-wise product of any two (complete) lattices.

PROPOSITION 3.6 (Lattice structure of $\mathcal{OP}$).
The set of orthogonal pairs forms a complete sub-lattice of the lattice $\mathcal{P}(\mathcal{SNT}) \times \mathcal{P}^{\mathrm{op}}(\mathcal{SNK})$, i.e. for $P, Q \in \mathcal{OP}$,

$$P \leq Q \quad \textit{iff} \quad (P)^{\mathsf{T}} \subseteq (Q)^{\mathsf{T}} \text{ and } (P)^{\mathsf{K}} \supseteq (Q)^{\mathsf{K}};$$

the join and meet of arbitrary sets $S \subseteq \mathcal{OP}$ are

$$\bigvee S \equiv \left(\bigcup_{P\in S} (P)^{\mathsf{T}}, \bigcap_{P\in S} (P)^{\mathsf{K}}\right) \qquad\qquad \bigwedge S \equiv \left(\bigcap_{P\in S} (P)^{\mathsf{T}}, \bigcup_{P\in S} (P)^{\mathsf{K}}\right),$$

and, explicitly, the empty joins and meets are

$$\bot \equiv (\emptyset, \mathcal{SNK}) \qquad\qquad \top \equiv (\mathcal{SNT}, \emptyset).$$

Orthogonal normal pairs. The other lattice we are interested in is the lattice $\mathcal{ONP}$ of what we call *orthogonal normal pairs*. These are orthogonal pairs that are made out at the outermost level of either introductions or eliminators. Logically speaking, they correspond to those proofs whose last derivation is a left or right operational rule; computationally, they would intuitively correspond to '(co-) normal forms' of a type. Orthogonal normal pairs inherit the lattice structure of $\mathcal{OP}$ but for the empty lub and glb that become $\bot \equiv (\emptyset, \mathcal{EK} \cap \mathcal{SNK})$ and $\top \equiv (\mathcal{IT} \cap \mathcal{SNT}, \emptyset)$.

PROPOSITION 3.7 (Lattice structure of $\mathcal{ONP}$).
The set $\mathcal{ONP}$ can be turned into a sub-lattice of $\mathcal{OP}$. It is complete with extrema given by

$$\bot = (\emptyset, \mathcal{EK} \cap \mathcal{SNK}) \quad \text{and} \quad \top = (\mathcal{IT} \cap \mathcal{SNT}, \emptyset).$$

LEMMA 3.8 (Saturation and orthogonality).
Let $T \subseteq \mathcal{SNT}$ and $K \subseteq \mathcal{SNK}$ be saturated sets of terms and co-terms, respectively, satisfying $[\rightsquigarrow](T \bullet K) \subseteq \mathcal{SN}$, then $T \bullet K \subseteq \mathcal{SN}$.

PROOF. We prove the result indirectly by proving instead that for any *finite* subsets $T' \subseteq T$ and $K' \subseteq K$ one has $T' \bullet K' \subseteq \mathcal{SN}$. Given that both T and K are equal to the union of their finite subsets (and these are preserved by the syntactic operation of cutting, $- \bullet -$) the result will follow. The proof for these will be by induction on the *sum of the depths of all possible reduction paths of all phrases* *in the sets T' and K'*. As the sets are finite and the (co-) terms terminating (by virtue of being in $\mathcal{SNT}$ or $\mathcal{SNK}$, as appropriate), we are guaranteed that this measure is either zero—and only head reductions apply—or strictly reducing when we execute a single step of parallel reduction.

For the zero case, we have that

$$
\begin{aligned}
& [\rightsquigarrow](T \bullet K) \subseteq \mathcal{SN} \text{ and } \left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' = \emptyset \subseteq \mathcal{SN} \\
& \text{and } T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right) = \emptyset \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\rightsquigarrow](T' \bullet K') \subseteq \mathcal{SN} \text{ and } \left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' \subseteq \mathcal{SN} \text{ and } T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right) \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\rightsquigarrow](T' \bullet K') \cup \left(\left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K'\right) \cup \left(T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right)\right) \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\overset{\rightsquigarrow}{\rightsquigarrow}](T' \bullet K') \subseteq \mathcal{SN} \\
\Longrightarrow\ & T' \bullet K' \subseteq \mathcal{SN}.
\end{aligned}
$$

The induction step is proved with measured aid of the induction hypothesis. Assume the sum of the depths of all reduction paths for terms and co-terms in T' and K' is non-zero; it therefore follows that either

$$
\left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' \quad \text{and} \quad T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right)
$$

both have smaller sum than $T' \bullet K'$, or the sum of depths of reduction for one of them is zero and for the other it is smaller that that of $T' \bullet K'$. Saturation now takes on a critical rôle; because of it we know that both $\left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right)$ and $\left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right)$ are subsets of T and K, and, since they are finite, the induction hypothesis can be applied to them. In any of the cases above, be it by the induction hypothesis or by virtue of not having any reducible terms, we have that

$$
\left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' \subseteq \mathcal{SN} \quad \text{and} \quad T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right) \subseteq \mathcal{SN},
$$

whence it follows that

$$
\begin{aligned}
& [\rightsquigarrow](T \bullet K) \subseteq \mathcal{SN} \text{ and } \left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' \subseteq \mathcal{SN} \text{ and } T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right) \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\rightsquigarrow](T' \bullet K') \subseteq \mathcal{SN} \text{ and } \left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K' \subseteq \mathcal{SN} \text{ and } T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right) \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\rightsquigarrow](T' \bullet K') \cup \left(\left([\overset{\rightsquigarrow}{\rightsquigarrow}](T')\right) \bullet K'\right) \cup \left(T' \bullet \left([\overset{\rightsquigarrow}{\rightsquigarrow}](K')\right)\right) \subseteq \mathcal{SN} \\
\Longrightarrow\ & [\overset{\rightsquigarrow}{\rightsquigarrow}](T' \bullet K') \subseteq \mathcal{SN} \\
\Longrightarrow\ & T' \bullet K' \subseteq \mathcal{SN}.
\end{aligned}
$$

□

LEMMA 3.9 (Preservation of orthogonality).
Let $T, T' \subseteq \mathcal{SNT}$, $K, K' \subseteq \mathcal{SNK}$ be saturated sets such that $T \bullet K \subseteq \mathcal{SN}$ and $T' \bullet K' \subseteq \mathcal{SN}$; then:

$$
\begin{array}{cc}
\langle T, T' \rangle \bullet \mathit{fst}[K] \subseteq \mathcal{SN} & \langle T, T' \rangle \bullet \mathit{snd}[K'] \subseteq \mathcal{SN} \\
i_1 \langle T \rangle \bullet [K, K'] \subseteq \mathcal{SN} & i_2 \langle T' \rangle \bullet [K, K'] \subseteq \mathcal{SN} \\
\multicolumn{2}{c}{\mathit{not}\langle K \rangle \bullet \mathit{not}[T] \subseteq \mathcal{SN}} \\
\lambda x.\left(T'\big|^{T/x}\right) \bullet (T @ K') \subseteq \mathcal{SN} & (T \# K') \bullet \mu\alpha.\left(K\big|^{K'/\alpha}\right) \subseteq \mathcal{SN} \\
a\langle T \rangle \bullet a[K] \subseteq \mathcal{SN} & e\langle T \rangle \bullet e[K] \subseteq \mathcal{SN}.
\end{array}
$$

TABLE 6 Interpretations of the second-order Dual Calculus in $\mathcal{ONP}$ and $\mathcal{OP}$.

$[\![T]\!](\gamma) : \mathcal{ONP}$	$[\![A \wedge B]\!](\gamma) = (\!|A|\!)(\gamma) \wedge (\!|B|\!)(\gamma)$
$[\![X]\!](\gamma) = \gamma(X)$	$[\![A \vee B]\!](\gamma) = (\!|A|\!)(\gamma) \vee (\!|B|\!)(\gamma)$
$[\![A \to B]\!](\gamma) = (\!|A|\!)(\gamma) \to (\!|B|\!)(\gamma)$	$[\![A - B]\!](\gamma) = (\!|A|\!)(\gamma) - (\!|B|\!)(\gamma)$
$[\![\neg A]\!](\gamma) = \neg(\!|A|\!)(\gamma)$	
$[\![\forall X . A]\!](\gamma) = \forall\left\{(\!|A|\!)(\gamma\,[X \mapsto N]) \,\middle|\, N \in \mathcal{ONP}\right\}$	$(\!|T|\!)(\gamma) : \mathcal{OP}$
$[\![\exists X . A]\!](\gamma) = \exists\left\{(\!|A|\!)(\gamma\,[X \mapsto N]) \,\middle|\, N \in \mathcal{ONP}\right\}$	$(\!|T|\!)(\gamma) = [\coprod]\left([\![T]\!](\gamma)\right)$

PROOF. Apart from the implicative and subtractive cases, by Lemma 3.1, all the sets parametrizing the cuts are saturated—and clearly strongly normalizing as well. For the other two cases, we appeal first to Theorem 3.5 for the saturated T' and K to show that $T'|^{T/x}$ and $K|^{K'/\alpha}$ are also saturated—and by definition strongly normalizing.

By Lemma 3.8, it suffices to show that after head-reduction the resulting cuts are strongly normalizing—e.g. using adjointness (Prop. 3.2):

$$[\rightsquigarrow]\left(\lambda x.\left(T'|^{T/x}\right) \bullet \left(T@K'\right)\right) = \left(T'|^{T/x}\,[T/x]\right) \bullet K' \subseteq T' \bullet K' \subseteq \mathcal{SN}$$

$$[\rightsquigarrow]\left(\left(T\#K'\right) \bullet \mu\alpha.\left(K|^{K'/\alpha}\right)\right) = T \bullet \left(K|^{K'/\alpha}\,[K'/\alpha]\right) \subseteq T \bullet K \subseteq \mathcal{SN}.$$

The remaining cases are even easier. □

Type actions. Pairing together the actions of the introductions and eliminations of a given type allows us to construct elements of $\mathcal{ONP}$ whenever we apply them to orthogonal pairs. These *type actions* are defined in Table 5.

PROPOSITION 3.10 (Orthogonality for actions).
For any $P, Q \in \mathcal{OP}$

$$P \wedge Q \in \mathcal{ONP} \qquad P \vee Q \in \mathcal{ONP} \qquad \neg P \in \mathcal{ONP}$$
$$P \to Q \in \mathcal{ONP} \qquad P - Q \in \mathcal{ONP}$$

For any $S \subseteq \mathcal{ONP}$

$$\forall S \in \mathcal{ONP} \qquad \exists S \in \mathcal{ONP}$$

PROOF. Using the lattice properties of $\mathcal{ONP}$ (Proposition 3.7), the proof reduces to repeated use of Lemma 3.9 to establish the orthogonality of the given sets. For that, we need the orthogonality and the saturation properties of the components of $\mathcal{OP}$. It also follows from the latter that they are saturated (Lemma 3.1). By construction, they are all made at the outermost level out of constructors/ eliminators. □

Orthogonal completion. Now that we have interpretations for the actions that construct values/co-values of a type in $\mathcal{ONP}$, we need to go the other way (as per diagram 1, above) to $\mathcal{OP}$, so that we also include (co-) variables and (co-) abstractions in our interpretations. So, for saturated orthogonal sets of values T and of co-values K, the term and co-term completions of T and K are respectively defined as:

$$[T](L) = \mathit{Var} \cup T \cup \bigcup_{\alpha \in \mathit{Covar}} \alpha.\left(\mathcal{SN}|^{L/\alpha}\right), \quad [K](U) = \mathit{Covar} \cup K \cup \bigcup_{x \in \mathit{Var}} x.\left(\mathcal{SN}|^{U/x}\right).$$

Due to the non-determinism associated with the reduction of abstractions, we need guarantee that all added (co-) abstractions are compatible not only with the initial set of values, but also with any (co-) abstractions that have been added in the process—and vice-versa. In other words, we need to iterate this process by taking the least fix-point in the complete lattice of subsets of terms–and then, from it, obtain the continuations:

$$\underline{(T \| K)} = \big(\mathit{lfp}[T] \circ [K]), [K]\,(\mathit{lfp}([T] \circ [K]))\,\big).$$

(In fact, as has been remarked elsewhere [3, 19], all one needs is *a* fix-point.) That the fix-point exists is a consequence of the next proposition; that it always yields an orthogonal pair when applied to elements $N \in \mathcal{ONP}$, is proven in Proposition 3.14—and in this case, we term it the *structural completion* of N.

PROPOSITION 3.11 (Contra-variance of co- and term completions).
For any $T \subseteq \mathcal{T}$ and $K \subseteq \mathcal{K}$ the functions

$$[T] : \mathcal{P}(\mathcal{K}) \to \mathcal{P}(\mathcal{T}) \quad \text{and} \quad [K] : \mathcal{P}(\mathcal{T}) \to \mathcal{P}(\mathcal{K})$$

are contra-variant under inclusion of sets. Consequently, the compositions

$$[T] \circ [K] : \mathcal{P}(\mathcal{T}) \to \mathcal{P}(\mathcal{T}) \quad \text{and} \quad [K] \circ [T] : \mathcal{P}(\mathcal{K}) \to \mathcal{P}(\mathcal{K})$$

are monotone (covariant) functions.

LEMMA 3.12
For any set of co-terms L (resp. of terms U), if T (resp. K) is saturated, then so is $[T]\,(L)$ (resp. $[K]\,(U)$).

PROOF. The set $\mathcal{SN}$ is saturated, hence, by Theorem 3.5, so is $\mathcal{SN}\big|^{L/\alpha}$ for any co-term set L, and, by Lemma 3.1, $\alpha.\left(\mathcal{SN}\big|^{L/\alpha}\right)$ is, therefore, also saturated. The set of variables *Var* is, trivially, saturated, as is T (by assumption); the union of these sets for any choice of co-variable in the abstraction

$$[T](L) = \mathit{Var} \cup T \cup \bigcup_{\alpha \in \mathit{Covar}} \alpha.\left(\mathcal{SN}\big|^{L/\alpha}\right)$$

is also saturated. (Likewise for the co-term closure.) □

LEMMA 3.13
For an arbitrary set of terms T and an arbitrary set of co-terms K, let $P = \underline{(T \| K)}$. The following equalities hold:

$$(P)^{\mathsf{T}} = [T]\left((P)^{\mathsf{K}}\right) \qquad (P)^{\mathsf{K}} = [K]\left((P)^{\mathsf{T}}\right).$$

And from these, it follows easily that

$$\mathit{Var}, T \subseteq (P)^{\mathsf{T}} \qquad \mathit{Covar}, K \subseteq (P)^{\mathsf{K}}.$$

PROPOSITION 3.14
Let $N \in \mathcal{ONP}$ be an orthogonal normal pair. Its structural completion is an orthogonal pair:

$$\underline{\|}N = \left(\underline{(N)^{\mathsf{T}} \parallel (N)^{\mathsf{K}}}\right) \in \mathcal{OP}.$$

3.3 Orthogonal interpretations

Interpretations. Given a type T and a finite mapping γ containing its free type variables, ftv(T), to $\mathcal{ONP}$—called the interpretation context—we define (Table 6) two interpretations, $(\!|T|\!)(\gamma)$ in orthogonal pairs and $[\![T]\!](\gamma)$ in orthogonal normal pairs, by mutual induction on the structure of T (bound variables are, as usual, taken fresh). They both satisfy the weakening and substitution properties. The extension of an interpretation context γ where a type-variable X is mapped to $N \in \mathcal{ONP}$ is denoted by $\gamma\,[X \mapsto N]$.

THEOREM 3.15 (Well-definedness).
For any DC type T and for any suitable interpretation context γ (i.e. finite dom(γ) and ftv$(T) \subseteq$ dom(γ)):

$$[\![T]\!](\gamma) \in \mathcal{ONP} \qquad \text{and} \qquad (\!|T|\!)(\gamma) \in \mathcal{OP}.$$

LEMMA 3.16
The two interpretations can be easily related in the following way:

$$Var, \left([\![T]\!](\gamma)\right)^{\mathsf{T}} \subseteq \left((\!|T|\!)(\gamma)\right)^{\mathsf{T}} \qquad Covar, \left([\![T]\!](\gamma)\right)^{\mathsf{K}} \subseteq \left((\!|T|\!)(\gamma)\right)^{\mathsf{K}}.$$

Second-order properties. In addition to being orthogonal, the two interpretations of Table 6 also satisfy the two (standard) properties of any second-order system: weakening and substitution. The first is paramount in the generalization rules (right-rule for universal quantification, left-rule for existential quantification), the second conflates type instantiation (substitution) at the syntactic level with instantiation of interpretation at the semantic level.

LEMMA 3.17 (Weakening).
Let $N \in \mathcal{ONP}$ and $Y \notin$ ftv(T):

$$[\![T]\!](\gamma\,[Y \mapsto N]) = [\![T]\!](\gamma) \qquad (\!|T|\!)(\gamma\,[Y \mapsto N]) = (\!|T|\!)(\gamma).$$

LEMMA 3.18 (Substitution).
Let T, T' be types, ftv$(T) - \{Y\} \subseteq$ dom(γ) and ftv$(T') \subseteq$ dom(γ):

$$[\![T\left[T'/Y\right]]\!](\gamma) = [\![T]\!](\gamma\left[Y \mapsto [\![T']\!](\gamma)\right])$$

$$(\!|T\left[T'/Y\right]|\!)(\gamma) = (\!|T|\!)(\gamma\left[Y \mapsto [\![T']\!](\gamma)\right]).$$

Syntactic lifting. The final results that we need relate the structure of the calculus—its operators—with the interpretation of types. Concretely, the former are ‘morphisms’ between the latter.

LEMMA 3.18 (Conservation).
The syntactic introduction and elimination operators of the second-order Dual Calculus conserve the

interpretations in the following sense:

$$\left\langle \left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}, \left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{T}} \right\rangle \subseteq \left(\llbracket A\wedge B\rrbracket(\gamma)\right)^{\mathsf{T}}$$

$$fst\left[\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llbracket A\wedge B\rrbracket(\gamma)\right)^{\mathsf{K}} \qquad snd\left[\left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llbracket A\wedge B\rrbracket(\gamma)\right)^{\mathsf{K}}$$

$$i_1\left\langle\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}\right\rangle \subseteq \left(\llbracket A\vee B\rrbracket(\gamma)\right)^{\mathsf{T}} \qquad i_2\left\langle\left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{T}}\right\rangle \subseteq \left(\llbracket A\vee B\rrbracket(\gamma)\right)^{\mathsf{T}}$$

$$\left[\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}}, \left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llbracket A\vee B\rrbracket(\gamma)\right)^{\mathsf{K}}$$

$$not\left\langle\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}}\right\rangle \subseteq \left(\llbracket \neg A\rrbracket(\gamma)\right)^{\mathsf{T}} \qquad not\left[\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}\right] \subseteq \left(\llbracket \neg A\rrbracket(\gamma)\right)^{\mathsf{K}}$$

$$\lambda x.\left(\left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{T}}\Big|^{\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}/x}\right) \subseteq \left(\llbracket A\to B\rrbracket(\gamma)\right)^{\mathsf{T}}$$

$$\left(\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}} @ \left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{K}}\right) \subseteq \left(\llbracket A\to B\rrbracket(\gamma)\right)^{\mathsf{K}}$$

$$\left(\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}} \# \left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{K}}\right) \subseteq \left(\llbracket A - B\rrbracket(\gamma)\right)^{\mathsf{T}}$$

$$\mu\alpha.\left(\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}}\Big|^{\left(\llbracket B\rrbracket(\gamma)\right)^{\mathsf{K}}/\alpha}\right) \subseteq \left(\llbracket A - B\rrbracket(\gamma)\right)^{\mathsf{K}}$$

and for $S = \left\{\llbracket A\rrbracket(\gamma\,[X\mapsto N]) \,\middle|\, N\in\mathcal{ONP}\right\}$ with X fresh for γ:

$$a\left\langle\bigcap_{P\in S}(P)^{\mathsf{T}}\right\rangle \subseteq \left(\llbracket\forall X\,.\,A\rrbracket(\gamma)\right)^{\mathsf{T}} \qquad a\left[\bigcup_{P\in S}(P)^{\mathsf{K}}\right] \subseteq \left(\llbracket\forall X\,.\,A\rrbracket(\gamma)\right)^{\mathsf{K}}$$

$$e\left\langle\bigcup_{P\in S}(P)^{\mathsf{T}}\right\rangle \subseteq \left(\llbracket\exists X\,.\,A\rrbracket(\gamma)\right)^{\mathsf{T}} \qquad e\left[\bigcap_{P\in S}(P)^{\mathsf{K}}\right] \subseteq \left(\llbracket\exists X\,.\,A\rrbracket(\gamma)\right)^{\mathsf{K}}.$$

LEMMA 3.20 (Conservation for abstractions).
The abstraction operation takes the interpretation for terms into a subset of the interpretation for co-terms—and vice-versa for co-abstractions:

$$x.\left(\mathcal{SN}\Big|^{\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}/x}\right) \subseteq \left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}} \qquad \alpha.\left(\mathcal{SN}\Big|^{\left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{K}}/\alpha}\right) \subseteq \left(\llbracket A\rrbracket(\gamma)\right)^{\mathsf{T}}.$$

3.4 Adequacy

Substitutions. The result of a (head) reduction is not necessarily composed of sub-phrases of the original cut; for abstractions, it depends on the result of a substitution. The proof of strong normalization cannot simply rest on a direct induction on the property of being strongly normalizing. We must additionally show that said property is invariant whenever we make some substitution that respects the interpretations of the given types.

Given typing (co-) contexts Γ and Δ, and an interpretation context γ containing all the free-variables in those contexts, we define a (bi-) substitution σ for Γ, Δ, γ as being a finite mapping of (co-) variables into the parametrized interpretation of their types in the contexts, i.e.

$$(x:T)\in\Gamma \Longrightarrow \sigma(x)\in\left(\llbracket T\rrbracket(\gamma)\right)^{\mathsf{T}}, \qquad (\alpha:T)\in\Delta \Longrightarrow \sigma(\alpha)\in\left(\llbracket T\rrbracket(\gamma)\right)^{\mathsf{K}}.$$

The application of a substitution to a phrase p is denoted $p[\sigma]$. With this in hand, we can express the adequate strong normalization theorem as follows.

THEOREM 3.21 (Adequacy).
Let t, k and c stand for terms, co-terms and cuts of the dual calculus. For any typing contexts and co-contexts Γ and Δ, s.t.

$$\Gamma \vdash t : T \mid \Delta \qquad \Gamma \mid k : T \dashv \Delta \qquad \Gamma \vdash c \dashv \Delta,$$

for any (suitable) interpretation context γ for Γ, Δ and T, and correspondingly suitable substitution σ, we have that

$$t[\sigma] \in \big((\!|T|\!)(\gamma)\big)^{\mathsf{T}}, \quad k[\sigma] \in \big((\!|T|\!)(\gamma)\big)^{\mathsf{K}} \quad \text{and} \quad c[\sigma] \in \mathcal{SN}.$$

PROOF. Formally, by rule induction on the typing trees. We show the cases for subtraction. Terms are handled straightforwardly using the induction hypothesis:

$$\begin{aligned} & t[\sigma] \in \big((\!|A|\!)(\gamma)\big)^{\mathsf{T}} \quad \text{and} \quad k[\sigma] \in \big((\!|B|\!)(\gamma)\big)^{\mathsf{K}} \\ \Longrightarrow\ & (t[\sigma]\#k[\sigma]) \in \big((\!|A - B|\!)(\gamma)\big)^{\mathsf{T}} \\ \Longrightarrow\ & ((t\#k))\,[\sigma] \in \big((\!|A - B|\!)(\gamma)\big)^{\mathsf{T}}. \end{aligned}$$

On the continuation side, we assume that co-variable α is chosen fresh everywhere. As the typing context on the assumption of the typing rule has an extra $\alpha : B$, we have that for any substitution σ for the conclusion and $k' \in \big((\!|B|\!)(\gamma)\big)^{\mathsf{K}}$, the substitution $\sigma\,[k'/\alpha]$ is in the conditions of the theorem, and, therefore the induction hypothesis yields:

$$k\,[\sigma\,[k'/\alpha]] = k[\sigma]\,[k'/\alpha] \in \big((\!|A|\!)(\gamma)\big)^{\mathsf{K}},$$

and also, because $Covar \subseteq \big((\!|B|\!)(\gamma)\big)^{\mathsf{K}}$,

$$k\,[\sigma\,[\alpha/\alpha]] = k[\sigma] \in \big((\!|A|\!)(\gamma)\big)^{\mathsf{K}} \subseteq \mathcal{SNK},$$

by definition of restriction, we conclude

$$k[\sigma] \in \big((\!|A|\!)(\gamma)\big)^{\mathsf{K}}\,|^{((\!|B|\!)(\gamma))^{\mathsf{K}}/\alpha},$$

and thereby

$$\begin{aligned} & \mu\alpha.(k[\sigma]) \in \mu\alpha.\Big(\big((\!|A|\!)(\gamma)\big)^{\mathsf{K}}\,|^{((\!|B|\!)(\gamma))^{\mathsf{K}}/\alpha}\Big) \\ \Longrightarrow\ & \mu\alpha.(k[\sigma]) \in \big((\!|A - B|\!)(\gamma)\big)^{\mathsf{K}} \\ \Longrightarrow\ & \mu\alpha.(k)\,[\sigma] \in \big((\!|A - B|\!)(\gamma)\big)^{\mathsf{K}}. \end{aligned}$$

□

COROLLARY 3.22 (Strong normalization).
Every well-typed phrase of DC is strongly normalizing.

4 Mendler induction

Having covered the first theme of the paper, classical logical in its Dual Calculus guise, let us focus in this section on the second theme we are exploring: Mendler induction. As the concept may be rather foreign, it is best to review it informally in the familiar functional setting.

Inductive definitions. Roughly speaking, an inductive definition of a function is one in which the function being defined can be used in its own definition provided that it is applied only to values of strictly smaller character than the input. The fix-point operator

$$\begin{aligned} &\mathit{fix} : \big((\mu X.F(X) \to A) \to \mu X.F(X) \to A\big) \to \mu X.F(X) \to A \\ &\mathit{fix}\, f\, x = f\ (\mathit{fix}\, f)\ x \end{aligned}$$

associated to the inductive type $\mu X.F(X)$ arising from a type scheme $F(X)$, clearly violates induction and indeed breaks strong normalization: one can feed it the identity function to yield a looping term. One may naively attempt to tame this behavior by considering the following modified fix-point operator

$$\begin{aligned} &\mathit{fix}' : \big((\mu X.F(X) \to A) \to F\big(\mu X.F(X)\big) \to A\big) \to \mu X.F(X) \to A \\ &\mathit{fix}'\, f\ (\mathit{in}\ x') = f\ (\mathit{fix}'\, f)\ x' \end{aligned}$$

in which, for the introduction $\mathit{in} : F\big(\mu X.F(X)\big) \to \mu X.F(X)$, one may regard x' as being of strictly smaller character than $\mathit{in}(x')$. Of course, this is still unsatisfactory as, for instance, we have the looping term $\mathit{fix}'\ (\lambda f.\ f \circ \mathit{in})$. The problem here is that the functional $\lambda f.\ f \circ \mathit{in} : (\mu X.F(X) \to A) \to F\big(\mu X.F(X)\big) \to A$ of which we are taking the fix-point takes advantage of the concrete type $F\big(\mu X.F(X)\big)$ of x' used in the recursive call.

Mendler induction. The ingenuity of Mendler induction is to ban such perversities by restricting the type of the functionals that the iterator can be applied to: these should not rely on the inductive type but rather be abstract—in other words, be represented by a fresh type variable X as in the typing below[4]:

$$\begin{aligned} &\mathit{mitr} : \big((X \to A) \to F(X) \to A\big) \to \mu X.F(X) \to A \\ &\mathit{mitr}\, f\ (\mathit{min}\, x) = f\ (\mathit{mitr}\, f)\ x \end{aligned}$$

for min the introduction $F\big(\mu X.F(X)\big) \to \mu X.F(X)$.

Note that if the type scheme $F(X)$ is endowed with a polymorphic mapping operation $\mathit{map}_F : (A \to B) \to F(A) \to F(B)$, every term $a : F(A) \to A$ has as associated catamorphism $\mathit{cata}(a) \equiv \mathit{mitr}\ \big(\lambda f.\ a \circ (\mathit{map}_F f)\big) : \mu X.F(X) \to A$. In particular, one has $\mathit{cata}(\mathit{map}_F\, \mathit{min}) : \mu X.F(X) \to F\big(\mu X.F(X)\big)$.

5 Dual calculus with Mendler induction

Mendler induction. We shall now formalize Mendler induction in the classical calculus of Section 2. Additionally, we shall also introduce its dual, Mendler co-induction. This requires type constructors, syntactic operations corresponding to the introductions and eliminations and their typing rules and reduction rules. These are summarized in Table 7. First, we take a type scheme $F(X)$ and represent its inductive type by $\mu X.F(X)$—dually, we represent the associated co-inductive type by $\nu X.F(X)$.

Syntax. As usual, the inductive introduction, $\mathit{min}\,\langle - \rangle$, witnesses that the values of the unfolding of the inductive type $F(\mu X\,.\,F(X))$ are injected in the inductive type $\mu X.F(X)$. It is in performing

[4]We note that the original presentation of this inductive operator [17] was in System F and, accordingly, the operator considered instead functionals of type $\forall X\,.\,(X \to A) \to F(X) \to A$. Cognoscenti will recognize that this type is the type-theoretic Yoneda reformulation $\forall X\,.\,(X \to A) \to T(X)$ of $T(A) = F(A) \to A$ for $T(X) = F(X) \to A$.

TABLE 7 Extension of the second-order Dual Calculus with Mendler induction.

Types

$$T := \ldots \mid \mu X.F(X) \mid \nu X.F(X)$$

Syntax

$$t := \ldots \mid \underbrace{\ldots \mid \mathit{min}\,\langle t\rangle \mid \mathit{mcoitr}_{r,x}\left\langle t, t'\right\rangle}_{\text{Introductions}} \mid \ldots$$

$$k := \ldots \mid \underbrace{\ldots \mid \mathit{mitr}_{\rho,\alpha}\left[k, k'\right] \mid \mathit{mout}[k]}_{\text{Eliminations}} \mid \ldots$$

Head reduction

$$\mathit{min}\,\langle t\rangle \bullet \mathit{mitr}_{\rho,\alpha}\,[k,l] \rightsquigarrow t \bullet k\left[\mu\alpha.\big(\mathit{mitr}_{\rho,\alpha}\,[k,\alpha]\big)/\rho\right][l/\alpha]$$

$$\mathit{mcoitr}_{r,x}\,\langle t,u\rangle \bullet \mathit{mout}[k] \rightsquigarrow t\left[\lambda x.\,\big(\mathit{mcoitr}_{r,x}\,\langle t,x\rangle\big)/r\right][u/x] \bullet k$$

Parallel reduction

$$t_0 \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} t_1 \Longrightarrow \begin{cases} \mathit{min}\langle t_0\rangle \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{min}\langle t_1\rangle \\ \mathit{mcoitr}_{r,x}\,\langle t_0, t\rangle \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{mcoitr}_{r,x}\,\langle t_1, t\rangle \\ \mathit{mcoitr}_{r,x}\,\langle t, t_0\rangle \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{mcoitr}_{r,x}\,\langle t, t_1\rangle \end{cases}$$

$$k_0 \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} k_1 \Longrightarrow \begin{cases} \mathit{mout}[k_0] \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{mout}[k_1] \\ \mathit{mitr}_{\rho,\alpha}[k, k_0] \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{mitr}_{\rho,\alpha}[k, k_1] \\ \mathit{mitr}_{\rho,\alpha}[k_0, k] \mathrel{\overset{\rightsquigarrow}{\rightsquigarrow}} \mathit{mitr}_{\rho,\alpha}[k_1, k] \end{cases}$$

Typing rules

$$\frac{\Gamma \vdash t{:}F(\mu X.F(X)) \mid \Delta}{\Gamma \vdash \mathit{min}\langle t\rangle{:}\mu X.F(X) \mid \Delta}$$

$$\frac{\Gamma \mid k{:}F(X) \dashv \Delta, \rho{:}X - A, \alpha{:}A \qquad \Gamma \mid l{:}A \dashv \Delta}{\Gamma \mid \mathit{mitr}_{\rho,\alpha}[k,l]{:}\mu X.F(X) \dashv \Delta}$$

$$(X \text{ not free in } \Gamma,\ \Delta,\ A)$$

$$\frac{x{:}A, r{:}A \to X, \Gamma \vdash t{:}F(X) \mid \Delta \qquad \Gamma \vdash u{:}A \mid \Delta}{\Gamma \vdash \mathit{mcoitr}_{r,x}\langle t,u\rangle{:}\nu X.F(X) \mid \Delta}$$

$$(X \text{ not free in } \Gamma,\ \Delta,\ A)$$

$$\frac{\Gamma \mid k{:}F(\nu X.F(X)) \dashv \Delta}{\Gamma \mid \mathit{mout}[k]{:}\nu X.F(X) \dashv \Delta}$$

induction that we consume values of inductive type and, hence, the *induction operator* (or *iterator*, or *inductor*), $\mathit{mitr}_{\rho,\alpha}\,[k,l]$ corresponds to an elimination. It is comprised of an iteration step k, an output continuation l and two distinct induction co-variables, ρ and α. We postpone the explanation of their significance for the section on reduction below, but note now that the iterator binds ρ and α in the iteration continuation but not in the output continuation, thus, e.g.

$$\big(\mathit{mitr}_{\rho,\alpha}\,[k,l]\big)\left[k'/\rho\right]\left[l'/\alpha\right] = \mathit{mitr}_{\rho,\alpha}\left[k, l\left[k'/\rho\right]\left[l'/\alpha\right]\right].$$

The co-inductive operators, $\mathit{mcoitr}_{r,x}\,\langle t,u\rangle$ and $\mathit{mout}[k]$, are obtained via dualization. In particular, the co-inductive eliminator, $\mathit{mout}[k]$, witnesses that the co-values k of type $F(\nu X.F(X))$ determine the 'proper' (i.e. those that are not abstractions) co-values of $\nu X.F(X)$.

Reduction. To reduce an inductive cut $\mathit{min}\,\langle t\rangle \bullet \mathit{mitr}_{\rho,\alpha}\,[k,l]$, we start by passing the unwrapped inductive value t to the induction step k. However, in the spirit of Mendler induction, the induction step must be instantiated with the induction itself *and*, because we are in a classical calculus, with the output continuation—this is where the parameter co-variables come into play. The first co-variable, ρ, receives the induction; the induction step may call this co-variable (using a cut) arbitrarily and it must also be able to capture the output of those calls—in other words, it needs to *compose this continuation* with other continuations; therefore, one needs to pass $\mu\alpha.\big(mitr_{\rho,\alpha}\,[k,\alpha]\big)$, the induction with the output continuation (subtractively) abstracted. The other co-variable, α, represents in k the output of the induction—which for a call $mitr_{\rho,\alpha}\,[k,l]$ is l.[5] For co-induction, we dualize—in particular, the co-inductive call expects the lambda-abstraction of the co-inductive step.

Typing. Lastly, we have the typing rules that force induction to be well founded. Recall that this was achieved in the functional setting by forcing the inductive step to take an argument of arbitrary instances of the type scheme $F(X)$. Here we do the same. In typing $mitr_{\rho,\alpha}\,[k,l]$ for $\mu X.F(X)$, we require k to have type $F(X)$ where X is a variable that appears nowhere in the derivation except in the (input) type of the co-variable ρ.

Example: naturals. Let us look at a concrete example: natural numbers under the abstraction prioritizing strategy. We posit a distinguished type variable $\mathcal{B}$ and from it construct the type $1 \equiv \mathcal{B} \vee \neg\mathcal{B}$, which is inhabited by the witness of the law of the excluded middle, $* \equiv \alpha.(i_2\,\langle not\langle x.\,(i_1\,\langle x\rangle \bullet \alpha)\rangle\rangle \bullet \alpha)$. The base type scheme for the naturals is $F(X) \equiv 1 \vee X$, and the naturals are then defined as $\mathcal{N} \equiv \mu X.F(X)$. Examples of values of this type are:

$$\text{zero} \equiv min\,\langle i_1\,\langle *\rangle\rangle\,, \quad \text{one} \equiv min\,\langle i_2\,\langle \text{zero}\rangle\rangle \;\text{ and }\; \text{two} \equiv min\,\langle i_2\,\langle \text{one}\rangle\rangle\,.$$

For any continuation k on $\mathcal{N}$, the successor ‘function’ is defined as the following continuation for $\mathcal{N}$

$$succk^k \equiv x.\,(min\,\langle i_2\,\langle x\rangle\rangle \bullet k) \qquad (x \notin \text{fv}(k)).$$

Example: addition. The above primitives are all we need to define addition of these naturals. The inductive step ‘return m for zero, or induct and then add one’ is encoded as

$$\mathbf{Step}^m_{\rho,\alpha} \equiv \big[x.\,(m \bullet \alpha), x.\,\big(\big(x\#succk^{\alpha}\big) \bullet \rho\big)\big]\,.$$

THEOREM 5.1
Let n and m stand for the encoding of two natural numbers and the encoding of their sum be (by abuse of notation) $n+m$. Under the abstraction prioritizing reduction rule,

$$n \bullet mitr_{\rho,\alpha}\,\big[\mathbf{Step}^m_{\rho,\alpha}, l\big] \rightsquigarrow^* (n+m) \bullet l.$$

PROOF. Inducting on n: for $n = 0$,

$$\begin{aligned} \text{zero} \bullet mitr_{\rho,\alpha}\,\big[\mathbf{Step}^m_{\rho,\alpha}, l\big] &\rightsquigarrow i_1\,\langle *\rangle \bullet \mathbf{Step}^m_{\rho,\alpha}\,\big[\mu\alpha.\big(mitr_{\rho,\alpha}\,\big[\mathbf{Step}^m_{\rho,\alpha}, \alpha\big]\big)\,/\rho\big]\,[l/\alpha] \\ &\rightsquigarrow * \bullet x.\,(m \bullet l) \\ &\rightsquigarrow m \bullet l\,; \end{aligned}$$

[5]One may wonder if the output continuation is strictly necessary. As outputs appear on the right of sequents, and the induction is already a left-rule, the only possible alternative would be to add a co-variable to represent it. However, under this rule the system would no longer be closed under substitutions [15].

for $n+1 = min\,\langle i_2\,\langle n\rangle\rangle$

$$\begin{aligned}
& min\,\langle i_2\,\langle n\rangle\rangle \bullet mitr_{\rho,\alpha}\left[\mathbf{Step}^m_{\rho,\alpha}, l\right] \\
\rightsquigarrow\ & i_2\,\langle n\rangle \bullet \mathbf{Step}^m_{\rho,\alpha}\left[\mu\alpha.\left(mitr_{\rho,\alpha}\left[\mathbf{Step}^m_{\rho,\alpha}, \alpha\right]\right)/\rho\right][l/\alpha] \\
\rightsquigarrow\ & n \bullet x.\left(\left(x\#succk^l\right) \bullet \mu\alpha.\left(mitr_{\rho,\alpha}\left[\mathbf{Step}^m_{\rho,\alpha}, \alpha\right]\right)\right) \\
\rightsquigarrow\ & \left(n\#succk^l\right) \bullet \mu\alpha.\left(mitr_{\rho,\alpha}\left[\mathbf{Step}^m_{\rho,\alpha}, \alpha\right]\right) \\
\rightsquigarrow\ & n \bullet mitr_{\rho,\alpha}\left[\mathbf{Step}^m_{\rho,\alpha}, succk^l\right] \\
\rightsquigarrow\ & (n+m) \bullet succk^l \qquad \text{(IH)} \\
\rightsquigarrow\ & min\,\langle i_2\,\langle n+m\rangle\rangle \bullet l.
\end{aligned}$$

□

Notice how the inductive operator works by accumulating future actions in the continuation parameter as it consumes the value—this is characteristic of this style of programming.

Splitting the naturals. Let us take a slight generalization of the naturals, $\mu X\,.\,A \vee X$, where 1 is replaced by some fixed type A (not containing X free) and zero can be parametrized by a term of that type instead of the fixed witness $*$,

$$zero\langle t\rangle \equiv min\,\langle i_1\,\langle t\rangle\rangle \qquad succ\,\langle t\rangle \equiv min\,\langle i_2\,\langle t\rangle\rangle\,.$$

Another well-known operation on naturals is the witness $split : \mathbb{N} \to \mathbb{N} \vee \mathbb{N}$ of the partition of naturals between evens and odds. This is an inductive function that tells of a natural number not only if it is even or odd but also which *n*th even or odd it is. Since it consumes a natural, it will be a continuation; to simulate returning, we also must parametrize it with a continuation on $\mathbb{N} \vee \mathbb{N}$ with the relevant component being called correspondingly:

$$split\,[k] \equiv mitr_{\rho,\alpha}\left[[\mathbf{base}^\alpha, \mathbf{ind}^\alpha_\rho], k\right],$$

where

$$\begin{aligned}
\mathbf{base}^\alpha &\equiv x.\,(i_1\,\langle zero\langle x\rangle\rangle \bullet \alpha) \\
\mathbf{ind}^\alpha_\rho &\equiv x.\,((x\#[y.\,(i_2\,\langle y\rangle \bullet \alpha), y.\,(i_1\,\langle succ\,\langle y\rangle\rangle \bullet \alpha)]) \bullet \rho)\,.
\end{aligned}$$

PROPOSITION 5.2
Define

$$succ^n\,\langle a\rangle \equiv \underbrace{\text{succ}\langle\text{succ}\langle\ldots}_{n\text{ times}}\langle zero\langle a\rangle\rangle\rangle\rangle.$$

under the abstraction prioritizing reduction strategy, for any pairing of continuations $[e, o]$

$$succ^m\,\langle a\rangle \bullet split\,[[e,o]] \rightsquigarrow^* \begin{cases} succ^n\,\langle a\rangle \bullet e & \text{if } m = 2n \\ succ^n\,\langle a\rangle \bullet o & \text{if } m = 2n+1. \end{cases}$$

Co-induction 'via' induction. The example above is more than simply an arithmetic curiosity. If we straightforwardly take the syntactic dual—replacing induction with co-induction, disjunctions with conjunctions and inputs with outputs—we arrive at the following definitions:

$$head[k] \equiv mout[fst[k]] \qquad tail[k] \equiv mout[snd\,[k]]$$

$$merge\,\langle p\rangle \equiv mcoitr_{r,x}\,\langle\langle \mathbf{cobase}^x, \mathbf{coind}^x_r\rangle, p\rangle,$$

where

$$\mathbf{cobase}^x \equiv \alpha.(x \bullet fst[head[\alpha]])$$
$$\mathbf{coind}^x_r \equiv \alpha.(r \bullet (\langle\beta.\,(x \bullet snd\,[\beta])\,,\beta.\,(x \bullet fst[tail[\beta]])\rangle @\alpha))\,.$$

As their names suggest, they form the basis of the merging of co-inductive streams of some type A [14]. Their associated co-inductive type is $\nu X\,.A \wedge X$. Going forward, dualizing Prop. 5.2 as well, we can see this operator satisfies the specification we would expect from the merging operator: if we try to take out (read, pass to a continuation k) the element at position $2n$ (zero based) of the merged stream, we get element n of the first stream, and if we try to take out the element at position $2n+1$, we get the element at position n of the second stream.

PROPOSITION 5.3
Define

$$tail^n[k] \equiv \underbrace{\text{tail[tail[}\ldots}_{n \text{ times}}[head[k]]]],$$

under the co-abstraction prioritizing reduction, for any pair of streams $\langle a, b\rangle$

$$merge\,\langle\langle a, b\rangle\rangle \bullet tail^m[k] \rightsquigarrow^* \begin{cases} a \bullet tail^n[k] & \text{if } m = 2n \\ b \bullet tail^n[k] & \text{if } m = 2n+1. \end{cases}$$

PROOF. Either directly or by dualizing Prop. 5.2. □

It may seem odd that we can encode infinite streams in a language that is—as we shall shortly see—strongly normalizing. The strict duality of the Dual Calculus makes it possible to re-frame any co-inductive problem into a more familiar inductive one. In this case, induction is strongly normalizing because we ever only apply it to finite values; conversely, co-inductive values may be 'infinite' but can only ever be analyzed using finite sequences of $mout[-]$ operations.

6 Strong normalization for Mendler induction

We now come to the main contribution of the paper: the extension of the orthogonal pairs interpretation of the second-order Dual Calculus (Section 3) to Mendler induction—and the proof, thereby, that the extension is also strongly normalizing.

6.1 Sets of syntax

Set and lattice structure. The extension begins with the reformulation of the sets $\mathcal{T}$, $\mathcal{K}$ and $\mathcal{C}$, $\mathcal{SNT}$, $\mathcal{SNK}$ and $\mathcal{SN}$, $\mathcal{IT}$ and $\mathcal{EK}$ so that they accommodate the (co-) inductive operators. Modulo these changes, the definitions of $\mathcal{OP}$ and $\mathcal{ONP}$ remain the same, and so do the actions for propositional and second-order types and the orthogonal completion, $\perp\!\!\!\perp$. All that remains is to give suitable definitions for the (co-) inductive actions and the interpretations of (co-) inductive types. As before, we lift syntactic operators to the level of sets by taking the image of their actions on phrases, confusing the notation for both. These operators preserve saturation.

LEMMA 6.1
Let T and U be saturated sets of terms, K and L be saturated sets of co-terms. Inductive and co-inductive terms and co-terms built out of them and the inductive and co-inductive introductions and eliminations are saturated:

$$[\precapprox](\mathit{min}\,\langle T\rangle) \subseteq \mathit{min}\,\langle T\rangle \qquad [\precapprox]\left(\mathit{mitr}_{\rho,\alpha}\,[K,L]\right) \subseteq \mathit{mitr}_{\rho,\alpha}\,[K,L]$$

$$[\precapprox]\left(\mathit{mcoitr}_{r,x}\,\langle T,U\rangle\right) \subseteq \mathit{mcoitr}_{r,x}\,\langle T,U\rangle \qquad [\precapprox](\mathit{mout}[K]) \subseteq \mathit{mout}[K].$$

Inductive restrictions. The reduction rule for Mendler induction is unlike any other of the calculus. When performing an inductive step for $\mathit{mitr}_{\rho,\alpha}\,[k,l]$, the bound variable ρ will be only substituted by one specific term, namely $\mu\alpha.\left(\mathit{mitr}_{\rho,\alpha}\,[k,\alpha]\right)$. One needs a different kind of restriction to encode this invariant: take K and L to be sets of co-terms (intuitively, where the inductive step and output continuation live) and define the inductive restriction by

$$K/^{\rho}_{\alpha}L \equiv \left\{ k \in \mathcal{SNK} \,\middle|\, \text{for all } l \in L,\ k\left[\mu\alpha.\left(\mathit{mitr}_{\rho,\alpha}\,[k,\alpha]\right)/\rho\right][l/\alpha] \in K \right\},$$

and also for co-induction, for sets of terms T and U,

$$T/^{r}_{x}U \equiv \left\{ t \in \mathcal{SNT} \,\middle|\, \text{for all } u \in U,\ t\left[\lambda x.\left(\mathit{mcoitr}_{r,x}\,\langle t,x\rangle\right)/r\right][u/x] \in T \right\}.$$

LEMMA 6.2 (Saturation for Mendler restrictions).
Let T be a set of terms, and K a set of co-terms—both of them saturated—and let U and L be any set of terms and any set of co-terms, respectively. For any (distinct) variables r and x, and co-variables ρ and α,

$$K/^{\rho}_{\alpha}L \text{ and } T/^{r}_{x}U$$

are saturated.

LEMMA 6.3 (Preservation of (head) orthogonality).
Take $T, U \subseteq \mathcal{SNT}$ to be saturated sets of terms, $K, L \subseteq \mathcal{SNK}$ to be saturated sets of co-terms and assume that $T \bullet K \subseteq \mathcal{SN}$; it then follows that

$$[\rightsquigarrow]\left(\mathit{min}\,\langle T\rangle \bullet \mathit{mitr}_{\rho,\alpha}\left[K/^{\rho}_{\alpha}L, L\right]\right) \subseteq \mathcal{SN} \qquad (\rho \neq \alpha \in \mathit{Covar})$$

$$[\rightsquigarrow]\left(\mathit{mcoitr}_{r,x}\left\langle T/^{r}_{x}U, U\right\rangle \bullet \mathit{mout}[K]\right) \subseteq \mathcal{SN} \qquad (r \neq x \in \mathit{Var}).$$

LEMMA 6.4 (Preservation of orthogonality).
Take $T, U \subseteq \mathcal{SNT}$ to be saturated sets of strongly normalizing terms, $K, L \subseteq \mathcal{SNK}$ to be saturated sets of strongly normalizing co-terms and assume that $T \bullet K \subseteq \mathcal{SN}$; it then follows that

$$\mathit{min}\,\langle T\rangle \bullet \mathit{mitr}_{\rho,\alpha}\left[K/^{\rho}_{\alpha}L, L\right] \subseteq \mathcal{SN} \qquad (\rho \neq \alpha \in \mathit{Covar})$$

$$\mathit{mcoitr}_{r,x}\left\langle T/^{r}_{x}U, U\right\rangle \bullet \mathit{mout}[K] \subseteq \mathcal{SN} \qquad (r \neq x \in \mathit{Var}).$$

6.2 Orthogonal pairs

Mendler pairing. Combining the inductive restriction with the inductive introduction/elimination set operations, we can easily create orthogonal normal pairs—much as we did for the propositional actions—from two given orthogonal pairs: one intuitively standing for the interpretation of $F(\mu F.F(X))$ and the other for the output type. However, the interpretation of the inductive type should not depend on a specific choice of output type but should accept *all* instantiations of output, as well as *all* possible induction co-variables; model-wise this corresponds to taking a meet over all possible choices for the parameters:

$$\mathrm{MuP}(P) = \bigwedge_{\substack{Q\in\mathcal{OP}\\ \rho\neq\alpha\in Covar}} \left(min\left\langle (P)^{\mathsf{T}}\right\rangle, mitr_{\rho,\alpha}\left[(P)^{\mathsf{K}}/^{\rho}_{\alpha}(Q)^{\mathsf{K}}, (Q)^{\mathsf{K}}\right]\right) \in \mathcal{ONP},$$

and similarly for its dual:

$$\mathrm{NuP}(P) = \bigvee_{\substack{Q\in\mathcal{OP}\\ r\neq x\in Var}} \left(mcoitr_{r,x}\left\langle (P)^{\mathsf{T}}/^{r}_{x}(Q)^{\mathsf{T}}, (Q)^{\mathsf{T}}\right\rangle, mout\left[(P)^{\mathsf{K}}\right]\right) \in \mathcal{ONP}.$$

Monotonization. The typing constraints on Mendler induction correspond—model-wise—to a monotonization step. This turns out to be what we need to guarantee that an inductive type can be modeled by a least fix-point; without this step, the interpretation of a type scheme would be a function on complete lattices that would not necessarily be monotone. There are two possible universal ways to induce monotone endo-functions from a given endo-function f on a complete lattice: the first one, $\lceil f\rceil$, we call the monotone extension and use it for inductive types; the other one, the monotone restriction $\lfloor f\rfloor$, will be useful for co-inductive types. Their definitions[6] are:

$$\lceil f\rceil\, x \equiv \bigvee_{y\leq x} f y \qquad \text{and} \qquad \lfloor f\rfloor\, x \equiv \bigwedge_{x\leq y} f y.$$

They are, respectively, the least monotone function above and the greatest monotone function below f. Necessarily, by Tarski's fix-point theorem, they both have least and greatest fix-points, i.e. we have $lfp\,(\lceil f\rceil)$ and $gfp\,(\lfloor f\rfloor)$.

LEMMA 6.5 (Characterization of monotonizations).
If f is an arbitrary endo-function on a complete lattice L then its monotone extension, $\lceil f\rceil : L\to L$, is the least monotone function above f while its monotone restriction $\lfloor f\rfloor : L\to L$ is the greatest monotone function below f.

COROLLARY 6.6
The monotone extension $\lceil f\rceil : \mathcal{ONP}\to\mathcal{ONP}$ and the monotone restriction $\lfloor f\rfloor : \mathcal{ONP}\to\mathcal{ONP}$ of any endo-function f on orthogonal normal pairs both have fix-point points. Further, the sets of their respective fix-points form themselves sub-lattices of $\mathcal{ONP}$, with greatest and least elements.

Inductive actions. Combining the above ingredients, one can define the actions corresponding to inductive and to co-inductive types. They are parametrized by functions $f : \mathcal{ONP}\to\mathcal{OP}$,

$$\mu f \equiv lfp\,(\lceil \mathrm{MuP}\circ f\rceil) \in \mathcal{ONP} \quad \text{and} \quad \nu f \equiv gfp\,(\lfloor \mathrm{NuP}\circ f\rfloor) \in \mathcal{ONP}.$$

THEOREM 6.7 (Orthogonality for actions).
For any arbitrary function $f : \mathcal{ONP}\to\mathcal{OP}$,

$$\mu f \in \mathcal{ONP} \quad \text{and} \quad \nu f \in \mathcal{ONP}.$$

[6]Cognoscenti will recognize that they are point-wise Kan extensions.

PROOF. For $f : \mathcal{ONP} \to \mathcal{OP}$, both $\mathrm{MuP} \circ f$ and $\mathrm{NuP} \circ f$ are functions $\mathcal{ONP} \to \mathcal{ONP}$, so Corollary 6.6 witnesses the existence of the claimed fix-points for

$$\lceil \mathrm{MuP} \circ f \rceil \quad \text{and} \quad \lfloor \mathrm{NuP} \circ f \rfloor .$$

□

6.3 *Orthogonal interpretations*

Interpretations. The normal interpretations for (co-) inductive types associated to some type-scheme $F(X)$ given a (suitable) interpretation context gamma is

$$[\![\mu X.F(X)]\!](\gamma) = \mu(\llparenthesis F(X) \rrparenthesis(\gamma\,[X \mapsto -]))$$
$$[\![\nu X.F(X)]\!](\gamma) = \nu(\llparenthesis F(X) \rrparenthesis(\gamma\,[X \mapsto -]))$$

while the respective *orthogonal interpretations* are as before. These interpretations also satisfy the weakening and substitution properties.

THEOREM 6.8 (Well-definedness).
For any type T of the Mendler-inductive extension of DC and for any suitable interpretation context γ ($\mathrm{dom}(\gamma)$ finite and $\mathrm{ftv}(T) \subseteq \mathrm{dom}(\gamma)$),

$$[\![T]\!](\gamma) \in \mathcal{ONP} \qquad \text{and} \qquad \llparenthesis T \rrparenthesis(\gamma) \in \mathcal{OP}.$$

LEMMA 6.9
The two interpretations are still related (Lemma 3.16) by

$$Var, \left([\![T]\!](\gamma)\right)^{\mathsf{T}} \subseteq \left(\llparenthesis T \rrparenthesis(\gamma)\right)^{\mathsf{T}} \qquad \text{and} \qquad Covar, \left([\![T]\!](\gamma)\right)^{\mathsf{K}} \subseteq \left(\llparenthesis T \rrparenthesis(\gamma)\right)^{\mathsf{K}} .$$

LEMMA 6.10 (Weakening).
Let $N \in \mathcal{ONP}$ and $Y \notin \mathrm{ftv}(T)$:

$$[\![T]\!](\gamma\,[Y \mapsto N]) = [\![T]\!](\gamma) \qquad \llparenthesis T \rrparenthesis(\gamma\,[Y \mapsto N]) = \llparenthesis T \rrparenthesis(\gamma).$$

LEMMA 6.11 (Substitution).
Let T, T' be types, and $\mathrm{ftv}(T) - \{Y\} \subseteq \mathrm{dom}(\gamma)$ and $\mathrm{ftv}(T') \subseteq \mathrm{dom}(\gamma)$:

$$[\![T\left[T'/Y\right]]\!](\gamma) = [\![T]\!](\gamma\left[Y \mapsto [\![T']\!](\gamma)\right]) \qquad \llparenthesis T\left[T'/Y\right] \rrparenthesis(\gamma) = \llparenthesis T \rrparenthesis(\gamma\left[Y \mapsto [\![T']\!](\gamma)\right]).$$

6.4 *An inductive principle*

Classically reasoning about Mendler induction. The issue now centers around the co-term component of the interpretation of inductive types (and, dually, the term component of co-inductive types) and the conditions in which they create co-terms in the interpretation of inductive types. Roughly, the induction hypothesis of the adequacy theorem states that for any $N \in \mathcal{ONP}$, $m \in \left(\llparenthesis X - A \rrparenthesis\,(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}$, $l \in \left(\llparenthesis A \rrparenthesis\,(\gamma)\right)^{\mathsf{K}}$ and suitable substitution σ we have

$$k\,[\sigma[\rho \mapsto m][\alpha \mapsto l]] \in \left(\llparenthesis F(X) \rrparenthesis\,(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}, \tag{2}$$

and if we were to prove that $mitr_{\rho,\alpha}\,[k[\sigma], l[\sigma]] \in \left([\![\mu X.F(X)]\!]\,(\gamma)\right)^{\mathsf{K}}$ just by the fix-point property of the interpretation, we would need to have

$$(k[\sigma])\left[\mu\alpha.(mitr_{\rho,\alpha}\,[k[\sigma],\alpha])/\rho\right][l/\alpha] \in \left(\llparenthesis F(X) \rrparenthesis\,\left(\gamma\left[X \mapsto [\![\mu X.F(X)]\!]\,(\gamma)\right]\right)\right)^{\mathsf{K}}$$

for arbitrary $l \in \left(\llparenthesis A \rrparenthesis\,(\gamma)\right)^{\mathsf{K}}$. Instantiating Formula 2 to the case when N is the interpretation of our fix-point, $[\![\mu X.F(X)]\!]\,(\gamma)$, we see that in order to prove that $mitr_{\rho,\alpha}\,[k[\sigma], l[\sigma]] \in$

$\left(\llbracket \mu X.F(X) \rrbracket (\gamma)\right)^{\mathsf{K}}$ the interpretation of subtractive types requires that for any $l' \in \left(\llparenthesis A \rrparenthesis (\gamma)\right)^{\mathsf{K}}$ we already know $mitr_{\rho,\alpha}\left[k[\sigma], l'\right] \in \left(\llparenthesis \mu X.F(X) \rrparenthesis (\gamma)\right)^{\mathsf{K}}$—a circularity!

For ω-complete posets there is an alternative characterization of the least fix-point of a *continuous* function as the least upper bound of a countable chain. The completion operation $\perp\!\!\!\perp$ used in the definition of the $\mathcal{OP}$ interpretation is not continuous. However, classically, the least fix-point of any monotone function f on a complete lattice exists and lies somewhere in the transfinite chain [7]

$$d_{\alpha+1} = f\,(d_\alpha) \qquad \text{and} \qquad d_\lambda = \bigvee_{\alpha<\lambda} d_\alpha \;\; \text{(for limit } \lambda\text{)}$$

(and dually for co-induction).

Admissibility. This least fix point comes with a useful induction principle. Because we work with least upper and greatest lower bounds, we need to think in terms of their preservation. We say of a proposition (seen as a set) that it is admissible *iff* it satisfies:

1. Lub preservation: $S \subseteq \mathbf{P} \Longrightarrow \mathbf{P}\left(\bigvee S\right)$
2. Downward closure: $a \leq b$ and $\mathbf{P}(b) \Longrightarrow \mathbf{P}(a)$.

THEOREM 6.12 (Induction principle for monotone extensions of endofunctions).
Take L to be a complete lattice, let $f : L \to L$ be an endo-function (not necessarily a homomorphism) and $\mathbf{P}$ be an admissible property on (read, subset of) L. if

$$\mathbf{P}(a) \Longrightarrow \mathbf{P}(f\,a),$$

then property $\mathbf{P}$ holds for the least fix-point of its monotone extension,

$$\mathbf{P}\left(\mathit{lfp}\left(\lceil f \rceil\right)\right).$$

COROLLARY 6.13
Let $g : M \to M$ be an endo-function (not necessarily a homomorphism) and $\mathbf{P}$ be a property (alternatively read subset) on a complete lattice M for which:

1. Glb preservation: $S \subseteq \mathbf{P} \Longrightarrow \mathbf{P}\left(\bigwedge S\right)$
2. Upward closure: $a \leq b$ and $\mathbf{P}(a) \Longrightarrow \mathbf{P}(b)$
3. Function preservation: $\mathbf{P}(a) \Longrightarrow \mathbf{P}(g a)$.

The property holds for the greatest fix-point of its monotone restriction,

$$\mathbf{P}\left(\mathit{gfp}\left(\lfloor g \rfloor\,|\right)\right).$$

PROOF. Apply Theorem 6.12 to $L = M^{\mathrm{op}}$. □

6.5 Adequacy

As in Section 3, we establish strong normalization via conservation and adequacy results for the Mendler-inductive extension.

LEMMA 6.14 (Conservation).
For any typescheme F, type A and interpretation context γ suitable for both,

$$min\left\langle \left(\llparenthesis F\,(\mu X.F(X))\rrparenthesis(\gamma)\right)^{\mathsf{T}}\right\rangle \subseteq \left(\llparenthesis \mu X.F(X)\rrparenthesis(\gamma)\right)^{\mathsf{T}}$$

$$mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \left(\llparenthesis F(X)\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{K}} \middle| \begin{array}{l}\left(\llparenthesis X-A\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{K}}/\rho\\ \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}/\alpha\end{array}, \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llparenthesis \mu X.F(X)\rrparenthesis(\gamma)\right)^{\mathsf{K}}$$

$$mcoitr_{r,x}\left\langle\bigcap_{N\in\mathcal{ONP}} \left(\llparenthesis F(X)\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{T}} \middle| \begin{array}{l}\left(\llparenthesis A-X\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{T}}/r\\ \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{T}}/x\end{array}, \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{T}}\right\rangle \subseteq \left(\llparenthesis \nu X.F(X)\rrparenthesis(\gamma)\right)^{\mathsf{T}}$$

$$mout\left[\left(\llparenthesis F\,(\nu X.F(X))\rrparenthesis(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llparenthesis \nu X.F(X)\rrparenthesis(\gamma)\right)^{\mathsf{K}},$$

where X is fresh for type A.

PROOF. We focus here solely on the eliminator for inductive types. The challenge we are faced with is to recast the statement of conservation as a proposition within the confines of our induction principle (Theorem 6.12). Using the—by now familiar—fact that the terms (and co-terms) in the normal interpretation are included in the orthogonal one and defining the abbreviation

$$\mathcal{F}\,(N) \equiv \left(\llparenthesis F(X)\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{K}} \middle| \begin{array}{l}\left(\llparenthesis X-A\rrparenthesis(\gamma\,[X\mapsto N])\right)^{\mathsf{K}}/\rho;\\ \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}/\alpha\end{array} \subseteq \mathcal{SNK}$$

our goal can be re-framed as

$$mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}\,(N), \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\llbracket \mu X.F(X)\rrbracket(\gamma)\right)^{\mathsf{K}}$$

with the $\mathcal{ONP}$ interpretation of the inductive type being the least fixed point

$$lfp\left(\lceil \mathrm{MuP}\circ\left(\llparenthesis F(X)\rrparenthesis(\gamma\,[X\mapsto -])\right)\rceil\right).$$

Take then $\mathbf{P}$ to be

$$\mathbf{P}(N) \quad \textit{iff} \quad mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}\right] \subseteq (N)^{\mathsf{K}}.$$

To use the induction principle (Theorem 6.12), we need to prove that $\mathbf{P}$ is admissible (downward and least upper bound closed) and that it is preserved by $\mathrm{MuP}\circ\llparenthesis F(X)\rrparenthesis(\gamma\,[X\mapsto -])$.

Let $M \leq N \in \mathcal{ONP}$. For downward closure, by contra-variance of the order for the continuation side, it follows that $(N)^{\mathsf{K}} \subseteq (M)^{\mathsf{K}}$; whence, if $\mathbf{P}(N)$ holds, we have that

$$mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left(\llparenthesis A\rrparenthesis(\gamma)\right)^{\mathsf{K}}\right] \subseteq (N)^{\mathsf{K}} \subseteq (M)^{\mathsf{K}}$$

or, equivalently, $\mathbf{P}(M)$, as needed. For the least upper bound property, for $S \subseteq \mathbf{P}$ we consider the empty and non-empty cases separately. If $S = \emptyset$ then, trivially,

$$mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] \subseteq \mathcal{SNK} = (\bot)^{\mathsf{K}};$$

otherwise,

$$\begin{aligned}
&\text{for any } N \in S,\ mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] \subseteq (N)^{\mathsf{K}}\\
&\textit{iff } mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] \subseteq \bigcap_{N\in S} (N)^{\mathsf{K}}\\
&\textit{iff } mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] \subseteq \left(\bigvee_{N\in S} N\right)^{\mathsf{K}}\\
&\textit{iff } \mathbf{P}\left(\bigvee S\right).
\end{aligned}$$

The core of the proof lies in showing preservation of $\mathbf{P}$ by MuP $\circ\, (\!|F(X)|\!)(\gamma\,[X \mapsto -])$. Assume, to this end, that $N \in \mathcal{ONP}$ is such that $\mathbf{P}(N)$, whence we gather that

$$\begin{aligned}
& & mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] &\subseteq (N)^{\mathsf{K}}\\
&\textit{iff} & mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] &\subseteq \left(\llbracket X\rrbracket\,(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}\\
&\Longrightarrow & mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right] &\subseteq \left((\!|X|\!)(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}\\
&\textit{iff} & mitr_{\rho,\alpha}[\ldots,\alpha]\left[\left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}/\alpha\right] &\subseteq \left((\!|X|\!)(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}\\
& & \{\text{for any } N,\ \mathcal{F}(N) \subseteq \mathcal{SNK},\ \text{so also } mitr_{\rho,\alpha}\left[\bigcap\ldots,\ldots\right] &\subseteq \mathcal{SNK};\ \text{then by prop. 2}\}\\
&\textit{iff} & mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \alpha\right] &\subseteq \left((\!|X|\!)(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} |^{((\!|A|\!)(\gamma))^{\mathsf{K}}/\alpha}\\
&\textit{iff} & \mu\alpha.\left(mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \alpha\right]\right) &\subseteq \mu\alpha.\left(\left((\!|X|\!)(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} |^{((\!|A|\!)(\gamma))^{\mathsf{K}}/\alpha}\right)\\
&\textit{iff} & \mu\alpha.\left(mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \alpha\right]\right) &\subseteq \mu\alpha.\left((\ldots)^{\mathsf{K}} |^{((\!|A|\!)(\gamma[X\mapsto N]))^{\mathsf{K}}/\alpha}\right)\\
&\Longrightarrow & \mu\alpha.\left(mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \alpha\right]\right) &\subseteq \left(\llbracket X - A\rrbracket\,(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}\\
&\Longrightarrow & \mu\alpha.\left(mitr_{\rho,\alpha}\left[\bigcap\nolimits_{N\in\mathcal{ONP}} \mathcal{F}(N), \alpha\right]\right) &\subseteq \left((\!|X - A|\!)(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} \qquad (\star)
\end{aligned}$$

This will put us in a position to prove that the inductive calls that comprise the left-hand side of the inclusion satisfy the inductive restriction. To see this, take

$$mitr_{\rho,\alpha}\,[k,l] \in mitr_{\rho,\alpha}\left[\bigcap_{N\in\mathcal{ONP}} \mathcal{F}(N), \left((\!|A|\!)(\gamma)\right)^{\mathsf{K}}\right]$$

from which we have that $k \in \bigcap_{N \in \mathcal{ONP}} \mathcal{F}(N)$. Specifically for our assumed N that satisfies property **P**

$$k \in \left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} \left|\; \begin{array}{l} \left(⦇X - A⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} / \rho \\ \left(⦇A⦈(\gamma)\right)^{\mathsf{K}} / \alpha \end{array} \right. ,$$

and, from $\star$,

$$\mu\alpha.\left(mitr_{\rho,\alpha}\,[k,\alpha]\right) \in \left(⦇X - A⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}.$$

Combining these two observations with the definition of the substitution restriction yields

$$\begin{aligned} \text{for any } l' \in \left(⦇A⦈(\gamma)\right)^{\mathsf{K}}, k\left[\mu\alpha.\left(mitr_{\rho,\alpha}\,[k,\alpha]\right)/\rho\right]\left[l'/\alpha\right] \\ = k\left[\mu\alpha.\left(mitr_{\rho,\alpha}\,[k,\alpha]\right)/\rho, l'/\alpha\right] \\ \in \left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}}. \end{aligned}$$

As claimed, then, since, $k \in \mathcal{SNK}$

$$k \in \left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} /_{\alpha}^{\rho} \left(⦇A⦈(\gamma)\right)^{\mathsf{K}}.$$

From this we conclude that

$$\begin{aligned} & mitr_{\rho,\alpha}\left[\bigcap_{N \in \mathcal{ONP}} \mathcal{F}(N), \left(⦇A⦈(\gamma)\right)^{\mathsf{K}}\right] \\ & \subseteq mitr_{\rho,\alpha}\left[\left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} /_{\alpha}^{\rho} \left(⦇A⦈(\gamma)\right)^{\mathsf{K}}, \left(⦇A⦈(\gamma)\right)^{\mathsf{K}}\right] \\ & \subseteq \bigcup_{\substack{Q \in \mathcal{OP} \\ \rho \neq \alpha \in Covar}} mitr_{\rho,\alpha}\left[\left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)^{\mathsf{K}} /_{\alpha}^{\rho} (Q)^{\mathsf{K}}, (Q)^{\mathsf{K}}\right] \\ & = \left(\mathrm{MuP}\left(⦇F(X)⦈(\gamma\,[X \mapsto N])\right)\right)^{\mathsf{K}} \\ & = \left(\left(\mathrm{MuP} \circ ⦇F(X)⦈(\gamma\,[X \mapsto -])\right)\ N\right)^{\mathsf{K}} \end{aligned}$$

which is nothing but $\mathbf{P}\left(\left(\mathrm{MuP} \circ ⦇F(X)⦈(\gamma\,[X \mapsto -])\right)\ N\right)$. □

THEOREM 6.15 (Adequacy).
Let t be a term, k a co-term and c a cut of the second-order Dual Calculus extended with Mendler induction. For any contexts Γ and Δ, and types T for which they are well-typed,

$$\Gamma \vdash t : T \mid \Delta, \quad \Gamma \mid k : T \dashv \Delta, \text{ and } \quad \Gamma \vdash c \dashv \Delta,$$

any interpretation context γ and substitution σ for Γ and Δ validate

$$t[\sigma] \in \left(⦇T⦈(\gamma)\right)^{\mathsf{T}}, \quad k[\sigma] \in \left(⦇T⦈(\gamma)\right)^{\mathsf{K}}, \text{ and } \quad c[\sigma] \in \mathcal{SN}.$$

PROOF. By extending the argument of Theorem 3.21. For the iterator, the proof almost exactly boils down to proving that the induction hypothesis on $k[\sigma]$ (and $l[\sigma]$) implies the relevant pre-conditions of conservation (Lemma 6.14). A slight complication arises from the fact that the free type variables in A need not appear in the conclusion, and therefore, a context γ which satisfies the adequacy conditions for the conclusion, need not be suitable for the antecedents. To guarantee that that is the case, we extend any such γ with those type variables that appear in A but were not contemplated in γ. Denoting the set of these by $C = \mathrm{ftv}(A) - \mathrm{dom}(\gamma)$ we get a new context by assigning $\bot$[7] to those variables:

$$\gamma' = \gamma\,[C \mapsto \bot]\,.$$

We shall also need to consider a further extension that accounts for the extra (fresh) type variable X; this freshness can be used— by the weakening property—to assign an arbitrary $N \in \mathcal{ONP}$ to it. The set C is necessarily finite as it is bound by the free type variables that appear in the (finite) type A. By repeated applications of weakening, we have that any substitution σ in the adequacy conditions for the conclusion w.r.t. context γ is also valid for γ' as

$$x : T \in \Gamma \Longrightarrow \sigma(x) \in \big((\!|T|\!)(\gamma)\big)^{\mathsf{T}} = \big((\!|T|\!)(\gamma')\big)^{\mathsf{T}} = \big((\!|T|\!)(\gamma'\,[X \mapsto N])\big)^{\mathsf{T}},$$

$$\alpha : T \in \Delta \Longrightarrow \sigma(\alpha) \in \big((\!|T|\!)(\gamma)\big)^{\mathsf{K}} = \big((\!|T|\!)(\gamma')\big)^{\mathsf{K}} = \big((\!|T|\!)(\gamma'\,[X \mapsto N])\big)^{\mathsf{K}}.$$

For the return continuation l, we immediately get from the induction hypothesis and weakening that

$$l[\sigma] \in (\!|A|\!)(\gamma') = (\!|A|\!)(\gamma'\,[X \mapsto N])\,.$$

For the inductive step k, for fresh $\rho \neq \alpha \in \mathit{Covar}$ and any

$$k' \in (\!|X - A|\!)(\gamma'\,[X \mapsto N]) \quad \text{and} \quad l' \in (\!|A|\!)(\gamma') = (\!|A|\!)(\gamma'\,[X \mapsto N])$$

we have, then, that $\sigma[\rho \mapsto k'][\alpha \mapsto l']$ is a substitution for k in the conditions of the theorem for $\gamma'\,[X \mapsto N]$; therefore

$$k[\sigma][k'/\rho, l'/\alpha] = k[\sigma[\rho \mapsto k'][\alpha \mapsto l']] \in \big((\!|F(X)|\!)(\gamma'\,[X \mapsto N])\big)^{\mathsf{K}},$$

and in particular, this holds for $k' = \rho$ and $l' = \alpha$, so $k[\sigma] \in \mathcal{SNT}$. Then, from the definition of the restriction, and because the N above was chosen arbitrary,

$$k[\sigma] \in \bigcap_{N \in \mathcal{ONP}} \big((\!|F(X)|\!)(\gamma'\,[X \mapsto N])\big)^{\mathsf{K}} \left|\; \begin{array}{l} \big((\!|X - A|\!)(\gamma\,[X \mapsto N])\big)^{\mathsf{K}} / \rho \\ \big((\!|A|\!)(\gamma)\big)^{\mathsf{K}} / \alpha \end{array} \right. .$$

The conservation lemma then yields

$$\mathit{mitr}_{\rho,\alpha}\,[k, l]\,[\sigma] = \mathit{mitr}_{\rho,\alpha}\,[k[\sigma], l[\sigma]] \in \big((\!|\mu X\,.\,F(X)|\!)(\gamma')\big)^{\mathsf{K}},$$

which, by weakening ($\mathrm{ftv}(\mu X.F(X)) \subseteq \mathrm{dom}(\gamma)$), is equivalent to

$$\mathit{mitr}_{\rho,\alpha}\,[k, l]\,[\sigma] = \mathit{mitr}_{\rho,\alpha}\,[k[\sigma], l[\sigma]] \in \big((\!|\mu X\,.\,F(X)|\!)(\gamma)\big)^{\mathsf{K}}.$$

The (co-) injection cases are simple; the co-inductor is dealt similarly to the above. □

COROLLARY 6.16 (Strong normalization).
Every well-typed phrase of DC with Mendler induction is strongly normalizing.

[7] Indeed any arbitrary but fixed element of $\mathcal{ONP}$ would do.

7 Concluding remarks

We have investigated classical logic with Mendler induction, presenting a classical calculus with very general (co-) inductive types. Our work borrows from and generalizes systems based on Gentzen's *LK* under the Curry–Howard correspondence. Despite its generality, and as determined by means of a reducibility argument, our Dual Calculus with Mendler induction is well behaved in that its well-typed cuts are guaranteed to terminate. We expect—but have yet to fully confirm—that other models fit within our framework for interpreting Mendler induction; our prime example is based on inflationary fix-points like those used in complexity theory [8] that also apply to non-monotone interpretations.

It is known that *LK*-based calculi can encode various other calculi [6, 22]. Our calculus supports map operations for all positive (co-) inductive types. This may be used to encode Kimura and Tatsuta's [15] extension of the Dual Calculus with positive (co-) inductive types [9, ch. 5].

One avenue of research that remains unexplored is how one may extract proofs from within our system—in previous work, Berardi, et al. [4] showed how, embracing the non-determinism of reduction inherent in the Symmetric Lambda-Calculus (and also present in *DC*), one could express proof witnesses that behave like processes for a logic based on Peano arithmetic. A further direction would take these investigations into the realm of linear logic, where the connection with processes may be more salient.

Funding

This work was supported by the United Kingdom's Engineering and Physical Sciences Research Council [grant number EP/J500380/1 to M. D. C.].

Acknowledgements

Thanks to Anuj Dawar, Tim Griffin, Ohad Kammar, Andy Pitts and the participants and anonymous referees of LFCS 2016 for their comments and suggestions.

References

[1] A. Abel, R. Matthes and T. Uustalu. Iteration and coiteration schemes for higher-order and nested datatypes. *Theoretical Computer Science*, **333** 3–66, 2005.

[2] K. Y. Ahn and T. Sheard. A hierarchy of Mendler style recursion combinators: taming inductive datatypes with negative occurrences. In *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming*, ICFP '11, ed, pp. 234–246. ACM, New York, NY, USA, 2011.

[3] F. Barbanera and S. Berardi. A symmetric lambda calculus for classical program extraction. *Information and Computation*, **125,** 103–117, 1996.

[4] F. Barbanera, S. Berardi and M. Schivalocchi. 'Classical' programming-with-proofs in λ^{Sym}_{PA}. An analysis of non-confluence. In Martín Abadi and Takayasu Ito., eds. *Theoretical Aspects of Computer Software, vol. 1281 of Lecture Notes in Computer Science*, pp. 365–390. Springer Berlin Heidelberg, 1997.

[5] T. Crolard. A formulae-as-types interpretation of subtractive logic. *Journal of Logic and Computation*, **14,** 529–570, 2004.

[6] P. -L. Curien and H. Herbelin. The duality of computation. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming*, ICFP '00, ed, pp. 233–243. ACM, New York, NY, USA, 2000.

[7] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[8] A. Dawar and Y. Gurevich. Fixed point logics. *Bulletin of Symbolic Logic*, **8,** 65–88, 2002.

[9] M. D. Campos. *Mendler Induction and Classical Logic*. PhD Thesis, University of Cambridge, 2015 (submitted).

[10] D. Dougherty, S. Ghilezan, P. Lescanne and S. Likavec. Strong normalization of the dual classical sequent calculus. In Geoff Sutcliffe and Andrei Voronkov., eds. *Logic for Programming, Artificial Intelligence, and Reasoning, vol. 3835 of Lecture Notes in Computer Science*, pp. 169–183. SpringerBerlin Heidelberg, 2005.

[11] G. Gentzen. Investigations into logical deduction. *American Philosophical Quarterly*, **1** 288–306, 1964.

[12] B. Harper and M. Lillibridge. *ML with callcc is unsound*. *Post to TYPES mailing list*, 1991.

[13] C. -K. Hur, G. Neis, D. Dreyer and V. Vafeiadis. The power of parameterization in coinductive proof. In *Proceedings of the 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '13, ed, pp. 193–206. ACM, New York, NY, USA, 2013.

[14] B. Jacobs and J. Rutten. A tutorial on (co) algebras and (co) induction. *Bulletin-European Association for Theoretical Computer Science*, **62**, 222–259, 1997.

[15] D. Kimura and M. Tatsuta. Dual calculus with inductive and coinductive types. In Ralf Treinen., eds. *Rewriting Techniques and Applications, vol. 5595 of Lecture Notes in Computer Science*, pp. 224–238. SpringerBerlin Heidelberg, 2009.

[16] R. Matthes. *Extensions of System F by Iteration and Primitive Recursion on Monotone Inductive Types*. PhD Thesis, Ludwig-Maximilians Universität, May 1998.

[17] N. P. Mendler. Inductive types and type constraints in the second-order lambda calculus. *Annals of Pure and Applied Logic*, **51,** 159–172, 1991.

[18] J. C. Mitchell and G. D. Plotkin. Abstract types have existential type. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, **10,** 470–502, 1988.

[19] M. Parigot. Strong normalization of second order symmetric λ calculus. In Sanjiv Kapoor and Sanjiva Prasad., eds. *FST TCS 2000: Foundations of Software Technology and Theoretical Computer Science, vol. 1974 of Lecture Notes in Computer Science*, pp. 442–453. Springer-Berlin Heidelberg, 2000.

[20] N. Tzevelekos. Investigations on the dual calculus. *Theoretical Computer Science*, **360,** 289–326, 2006.

[21] T. Uustalu and V. Vene. Mendler-style inductive types, categorically. *Nordic Journal of Computing*, **6** 343, 1999.

[22] P. Wadler. Call-by-value is dual to call-by-name. In *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming*, ICFP '03, ed, pp. 189–201. ACM, New York, NY, USA, 2003.

Received 16 December 2019